# DIFFUSION AND INVASION PERCOLATION: A COUPLED MODEL TO SIMULATE THE FATE OF TRAPPED AND COMPRESSED GASES IN PARTIALLY-SATURATED LOW PERMEABILITY MEDIA

A Thesis
presented in partial fulfillment of requirements
for the degree of Masters of Engineering Science
in the Department of Geology and Geological Engineering
The University of Mississippi

by

ROSS A. BERRY

MAY 2016

ABSTRACT

A three-dimensional computer program written in FORTRAN 90 has been coupled with a modified macroscopic invasion percolation (MMIP3) model. We focus on the influence of diffusion on the fate of trapped and compressed gases in low-permeability soils and rocks. The MMIP3 model simulates invasion of a wetting fluid (water) into a medium characterized by a random field of spatially correlated water entry pressures and re-invasion of a non-wetting fluid (*e.g.*, $N_2$). We simulate invasion of the aqueous phase followed by dissolution and diffusion of the gas phase. In an example, we consider the wetting of low-permeability mudrocks following a flooding event. First, invasion of water is simulated. The invasion process is then stopped to allow dissolution and diffusion of the gas phase. The dissolution of the gas phase lowers the gas pressures in the gas clusters. Then, invasion is restarted. This process is repeated eight times until the invasion process reaches equilibrium. The cumulative time allowed for diffusion was 10,000 years. We find that the air cluster pressures are decreased by dissolution, with transport via diffusion, which decreased the average air pressure and increased saturation across the system by 10 % over the simulated time. The number of cluster cells doubled compared to simulations without diffusive losses. We attribute this to decreased cluster gas pressures, which increases the quantity of water invasion and splits larger continuous clusters into small separate clusters. The models developed here can be applied to understanding the impact of diffusion on trapped and compressed gases in partially-saturated media and can be applied for other scenarios, such as $CO_2$ leakage from injection boreholes at sequestration sites. We find that trapped gas

pressures are reduced by dissolution into the aqueous phase, with transport via diffusion leading to higher water saturation values and lower average air pressures in trapped regions.

TABLE OF CONTENTS

LIST OF FIGURES

# I. INTRODUCTION

In partially-saturated porous media, complicated wetted phase structures develop due to capillary and geologic heterogeneity, and the influence of buoyancy and viscous forces. These phase structures can directly impact the transport properties of a porous media (*Singh et al*., 2010). The presence of trapped and compressed gases in a thick sequence of low-permeability soil and rock influences infiltration rates (*McWhorter*, 1971), flow instabilities (*Wang et al*., 1998) and hydrologic transport properties (*Holocher et al*., 2003). The presence of excess residual air can convey information about past climatic and recharge conditions as well as aquifer characteristics (*Holocher et al.*, 2002). The soil air, initially at atmospheric pressure, can become compressed ahead of the wetting front during recharge events following flooding (*Navarro et al*., 2008). Infiltration into an unsaturated, low-permeability medium is an important problem due to environmental concerns associated with waste disposal in arid regions (*Glass et al*., 2004). Previous laboratory and field studies have indicated that dissolved gas transport in groundwater can be greatly affected by the presence of even small amounts of trapped gas or residual air in the pore space (*Donaldson*, 1998). Several attempts have been made to mathematically and experimentally simulate the complex effects of air entrapment on infiltration properties [*Morel-Seytoux and Khanji*, 1974; *Morel-Seytoux and Billica*, 1985; *Parlange and Hill*, 1976; *Sander et al*., 1988; *Ioannou et al.*, 2003], but the underlying physical processes affecting infiltration in the presence of entrapped air are still not fully understood conceptually (*Wang et al.*, 1998). We use the term 'trapped gas' to mean 'residual air' or 'entrapped air'.

Also, we use the term ‘cluster’ to represent contiguous regions occupied by gas with the same pressure, irrespective of elevation.

The present study couples a newly written three-dimensional (3D) diffusion code (DIFF_PERC) with an existing 3D modified macroscopic invasion percolation (MMIP3) code, written by *Singh et al*., (2010), to quantify the fate and longevity of entrapped and compressed gas. Diffusion is the principal mechanism in the interchange of gases between soils through water to the atmosphere (*Rolston*, 1986). Given the long time scale for most regional hydrologic events, we assume that dissolution of trapped air is limited by diffusion, not the kinetics of dissolution. A backwards in time, block-centered, integrated finite-difference approach is used to solve a 3D transient diffusion equation on a uniform grid. The model is capable of supporting two diffusion coefficients for water and a given soil type with $1^{st}$ and $2^{nd}$ type boundary conditions (*e.g.*, constant concentration or constant flux). The model has been verified by comparing the numerical results with analytical solutions for molecular diffusion and analogous transport parameters (*e.g*., heat transfer).

A combination of two different modeling approaches was applied to model trapped and compressed gases before and after water inundation. The MMIP3 code, used to simulate the imbibing fluid, incorporates capillary and buoyancy forces and allows for trapping, compression, and expansion of the gas phase. The MMIP3 model simulates the evolution of trapped and compressed gases following the flooding of a geologic unit. In baseline simulations, the MMIP3 model is allowed to evolve to a steady-state condition, but we halt the percolation code before it reaches a steady state. The pressure associated with each trapped zone of gas is then mapped as initial air concentrations for the diffusion code.

Diffusive losses from the trapped and compressed zones through the water to the atmosphere are then simulated for a user specified time. Following the diffusion step, the capillary pressures of the trapped and compressed gas clusters are adjusted to reflect the losses due to dissolution and diffusion. The MMIP3 code is then restarted from its previous state to allow the wetting front to advance and stopped again. This successive starting and stopping of the percolation code represents the real kinetic growth event that would occur in nature, that is, a wetting front pushes air downward, the gas diffuses into the aqueous solution and eventually back to the atmosphere, and the wetting front continues. Once steady state is reached, the MMIP3 model is restarted a final time with the water overburden removed. Dissolution will lead to reduced gas pressures and invasion of pores by water. If dissolution is minimal, removing the original water depth will decrease the water pressure and push water out of the pores by expansion. The overall objective of this study is to couple a newly written 3D diffusion code with an existing percolation program to assist in further understanding hydrologic transport properties in the presence of trapped and compressed gas in partially-saturated porous media, specifically low-permeability soils, under conditions that would arise in nature.

## I.I BACKGROUND

Entrapped air not only influences the gas composition but also the transport dynamics of dissolved gas components in groundwater systems (*Holocher et al*., 2003). Previous laboratory studies [*Wilson and Luthink,* 1963; *Peck,* 1965; *Adrian and Franzini,* 1966; *Latifi et al*., 1994] and field experiments [*Bodman,* 1937; *Dixon and Linden,* 1972; *Jalali-Farahni et al.,* 1993] have shown that compressed soil gas can lead to a substantial decrease in infiltration rates (*Wang et al*., 1998). Trapped gas can be created in an unconfined aquifer due to water level fluctuations caused by seasonal changes in aquifer recharge or discharge (*Fry,* 1995). During inundation of a porous medium, water imbibes into soils, trapping and compressing gas within zones of larger pores. While the region is underwater, entrapped air may move by dissolving into the water and diffusing to the surface of the water table where the aqueous concentration is in equilibrium with the atmosphere (*Fry,* 1995) and eventually back into the atmosphere (*Bloomsburg and Corey*, 1964) (Figure 1).

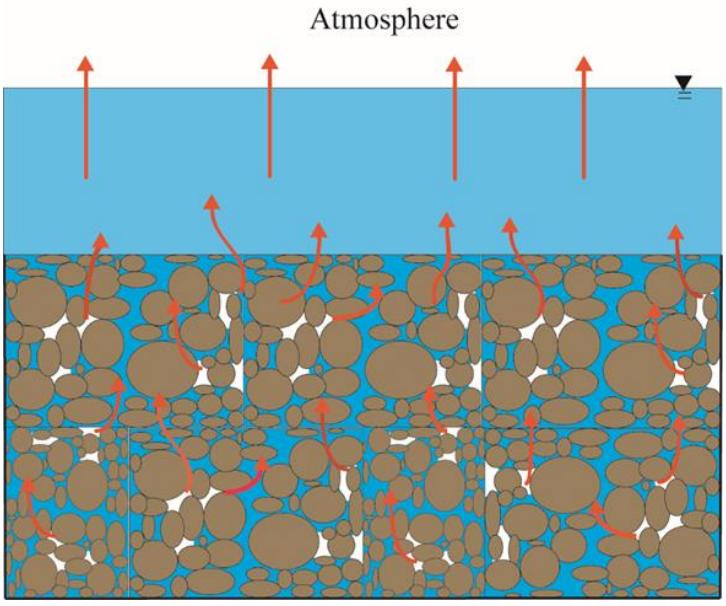



Figure 1. Conceptual model. Entrapped air (white) in a partially-saturated porous media with overburden of standing water (blue) and diffusion (red) to the atmosphere.

As pressures in the gas phase decrease from diffusive losses, the water content will increase. If the water pressure in the system is decreased (*e.g*., removing ponded water), remaining trapped and compressed gases can expand, decreasing the water content of the soil.

Normally, when liquid pressures are greater than atmospheric pressure, porous media are often assumed to be fully saturated, however, if there is entrapped air, the permeability of media to liquid may be only half that in completely saturated conditions (*Bloomsburg and Corey*, 1964). Possible mechanisms controlling excess air in a system are the partial diffusive degassing of the initial excess air with atmospheric pressures across the groundwater table, or the equilibration of a finite water volume with a finite air volume under increased pressures (*Holocher et al*., 2002).

Entrapped air may occur by water moving into an initially dry soil (*Bloomsburg and Corey,* 1964). Advancing wetting fronts push air downward and trap air in large pores. As the water pressure increases, zones of trapped air get compressed, leaving zones of interconnected compressed gas at depth. A complicated history of wetting and surface drying typically leads to these conditions (*e.g*., *Holt et al*., 2010). An analytical solution for diffusion of the gas phase (*Crank,* 1975) suggests that trapped gas can persist for millions of years in low-permeability rocks (*e.g*., the Dockum Group of West Texas) (Figure 2).

At the Waste Control Specialists site in West Texas, Triassic age mudrocks appear to have a trapped and compressed gas phase. Capillary pressures in mudrock can average 3 MPa (megapascal) while the capillary pressures in instrumented boreholes can average 0.3 MPa (*Holt et al*., 2008).

*Holt et al*., (2010) suggest a compressed air model can also easily account for discrepancies observed between core-based measurements of capillary pressure and those from instrumented boreholes and piezometers. They argue that when a borehole is drilled into an unsaturated rock containing a trapped and compressed gas phase, the gas phase pressure equilibrates with the atmospheric pressure, lowering capillary pressures (increasing the water potential). The rate of this equilibration depends primarily on the hydraulic conductivity of the medium.

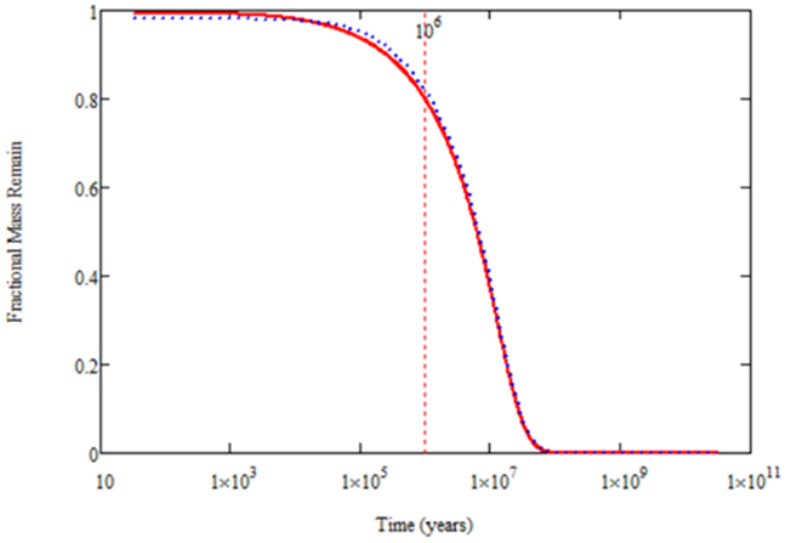



Figure 2. Dissolution of the air phase. A 100 meter column of soil exposed to the atmosphere (red) versus a 200 meter column of water over the soil (blue) with surface evaporation. Approximately 80% of the initial mass remains after 1 MY with a diffusion coefficient of $10^{-11}$ $m^2s^{-1}$ (*Crank*, 1975) and approximately 82% of the mass remains when the overlying water column is applied.

Our conceptual model is that during inundation by water, bubbles of soil gas are trapped in some pores and pressurized by the hydrostatic overburden and surface tension. This entrapped air does not completely dissolve over time, but instead reaches solubility equilibrium with the surrounding water under increased pressures (*Aeschbauch-Hertig et al*., 2001). Experimental studies have shown that residual excess air is formed in the presence of a progressively

dissolving gas phase, that is, the entrapped air component has an elemental composition similar to that of free atmospheric pressures (*Klump et al*., 2008). During fluctuations of the water table, the water phase is periodically replaced by soil air, which is assumed to be atmospheric for our model. The gas clusters are spatially fixed, since the capillary forces that trap the gas phase dominate any buoyancy or viscous forces in a porous medium, unless the pressure in the air phase exceeds the air-entry pressure.

## I.II INVASION PERCOLATION

Invasion percolation models (IPMs) simulate processes involving the flow of two immiscible fluids (*e.g*., $CO_2$ and water) in a porous media and are more favorable for revealing transport and flow parameters from actual complex phase distributions of soil parameters observed in the field than ordinary percolation and continuum models (*Singh et al*., 2010). Continuum models cannot capture the actual observed phase structure of transport in an unsaturated porous media as accurately as an IPM. In order to capture these complex flow parameters and behaviors, a porous medium can be virtually represented as an ordered lattice with assigned local-scale dependencies such as porosity, permeability, pore size distributions and other hydrologic parameters rather than micro-scale pore and pore throat distributions (*Kueper and McWhorter*, 1992).

Invasion percolation is a kinetic growth event that describes displacement of one fluid by another immiscible fluid at a constant flow rate. Ordinary percolation describes displacement at a constant applied pressure in an equilibrium system.

According to the dynamical rule in invasion percolation, an interface (water) will advance through a path of least resistance while the rule for ordinary percolation states that all interfaces will advance up to some predetermined threshold (applied pressure) of resistance in a porous medium. Invasion percolation involves a unique sequence of advances of the interface that determine whether or not a portion of a displaced fluid will become trapped. In contrast, the advancement of an interface at a given applied pressure and different time scales can lead to different trapping configurations (*Wilkinson and Willemsen*, 1983). The slow transport of fluid through a porous medium (*e.g*., water displacing air in a mudrock) is directly related to heterogeneities and hydrologic properties within that medium. IPMs are used to study imbibition, the process of a wetting fluid displacing a non-wetting fluid for replicating the transport path a fluid will take based on gravitational and capillary forces. Invasion percolation occurs within a network of connected pores where each pore (site) has an assigned random invasion pressure based on the type of soil or rock. Each lattice on a network of regular lattice sites represent areas characterized by different hydrologic properties (*e.g.*, hydraulic conductivity, capillary pressures, and porosity). A porous medium is traditionally represented by a network of pores (nodes) joined by narrower connecting throats. In an idealized porous medium, the network can be viewed as a lattice with the sites and bonds representing the pores and throats (*Ionnidis et al*., 1996). Percolation theory is strictly applicable only when flow rates are constant and infinitesimally small, viscous forces are completely neglected, and the system is driven by capillary forces (*Ionnadis et al*., 1996). When water displaces air at low flow rates, the viscous forces are completely dominated by the capillary forces at the air-water interface. The capillary forces control whether or not water will spontaneously displace air and are strongest at the narrowest places in the medium.

Therefore, if all the throats are smaller than all the pores, the water-air interface moves quickly through the throats, but get trapped after entering larger pores (*Wilkinson and Willemsen*, 1983). A simple theoretical model can be formulated where the motion of this water-air interface is represented by a series of discrete steps, where water displaces air from the smallest available pore or the greatest capillary drive. Specific pores are invaded with the invading phase at a boundary edge. The simulation of an IPM process requires following and keeping track of the water-air interface as it advances through the smallest pores available, in a series of steps.

An alternative to pore-scale lattice models are representative elementary volume (REV) macroscopic scale models. A REV approach that modifies the invasion percolation model by including buoyancy (gravitational) forces and viscous forces, although extremely small in comparison to capillary forces, is called a Macroscopic Invasion Percolation (MIP) model. The MMIP3 code is based macro modified invasion percolation reported by *Glass et al*., (2001) to simulate the unsaturated non-wetting fluid displacement of a wetting fluid. A MIP model can be used for visualizing invasion processes at discrete time steps and can be applied and altered for problems that require complex phase distributions and parameters. A MIP model differs from and IPM through the definition of macro-scale capillarity where individual pore throats and necks are not considered. Instead, a near pore-scale block is defined and characterized by a local spanning pressure ($P_s$) and invasion (breakthrough) pressure that represent the properties of the subscale network. Constructing a computer simulation of the displacement process requires a model domain discretized into an array of grid blocks with assigned spanning pressures and an invasion percolation algorithm that sorts the invadable blocks (cells), selects the block connected to the growing cluster with the lowest invasion pressure, and invades it. This process can be reversed for a non-wetting fluid replacing a wetting fluid (*e.g*., drainage).

As invasion advances, isolated pockets of gas (clusters) can form and be trapped by the invading fluid. Cutting off the gas in these clusters can lead to a rise in pressure above their initial atmospheric pressure. MIP generated phase structures can be mapped to transport properties that have constitutive relationships for hydrologic parameters such as pressure-saturation, permeability, and mass-transfer coefficients.

## I.III SCOPE

The objective of this work is to model the origin and fate of compressed and trapped gases within a low-permeability soil following flooding of water, specifically by the role of dissolution via diffusive transport, by coupling the existing MMIP3 model with the 3D diffusion code. Other subprograms were written to couple the diffusion code with the percolation code. Their construction and capabilities will be briefly mentioned. Superimposing the diffusion code with the percolation code can aid in testing the hypothesis of *Holt et al*. (2010) that trapped and compressed gas phase can survive over long periods of time (*e.g.,* 1MY) as well as revealing underlying transport processes.

## II. METHODS

The diffusion and percolation code coupling sequence (Figure 3), tasks performed by supplemental codes, and internal subroutines will be discussed in sequential order here.

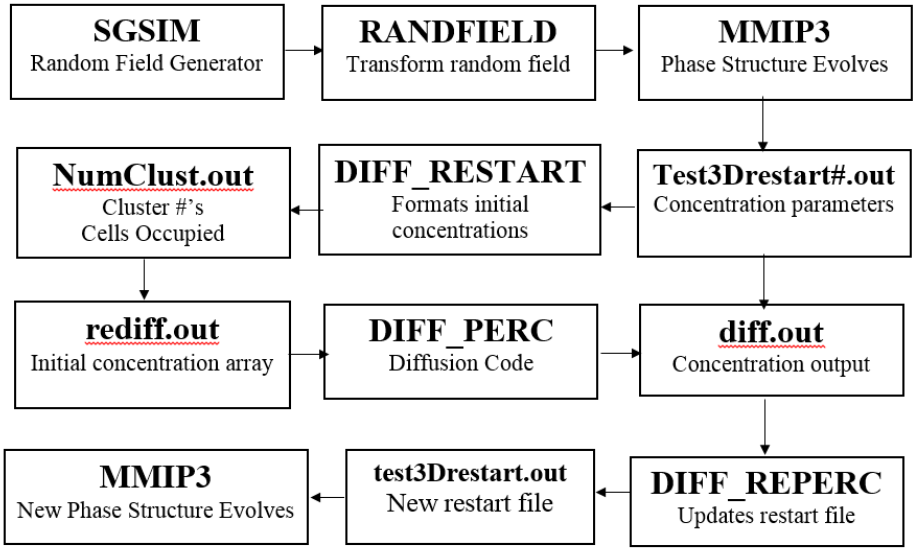



Figure 3. Coupling flow chart. Sequence of programs to incorporate the diffusion code DIFF_PERC with the MMIP3 percolation code.

Initial water entry (spanning) pressures are determined by randomly sampling the distribution of observed air-entry pressures from a low-permeability mudrock. Both correlated and uncorrelated random fields of spanning pressures were generated. The random fields used for the initial MMIP3 compilation may be generated over a grid equal to or bigger than the active grid used for the percolation simulation.

Correlated random fields (one random variable for one grid block) are generated over all cells in the active model domain to simulate the water entry pressures. These values represent a random field (isotropic/anisotropic) with a prescribed correlation structure. The random field files are generated using the code SGSIM from GSLIB90 software (*Deutsch et al*., 1998). The random fields reflect a 4 layer correlated anisotropic domain with an exponential covariance structure. Air-entry (spanning) pressures follow a log-normal distribution with geometric mean values of 2.2 and 1.8 MPa (log means of 8.65 and 5.95) and log variances of 2.7 and 1.1 for clay and sand, respectively (Figure 4). These statistical values were obtained from site specific air-entry pressure experiments and subsurface discontinuity mapping reports (*Holt et al.*, 2011).

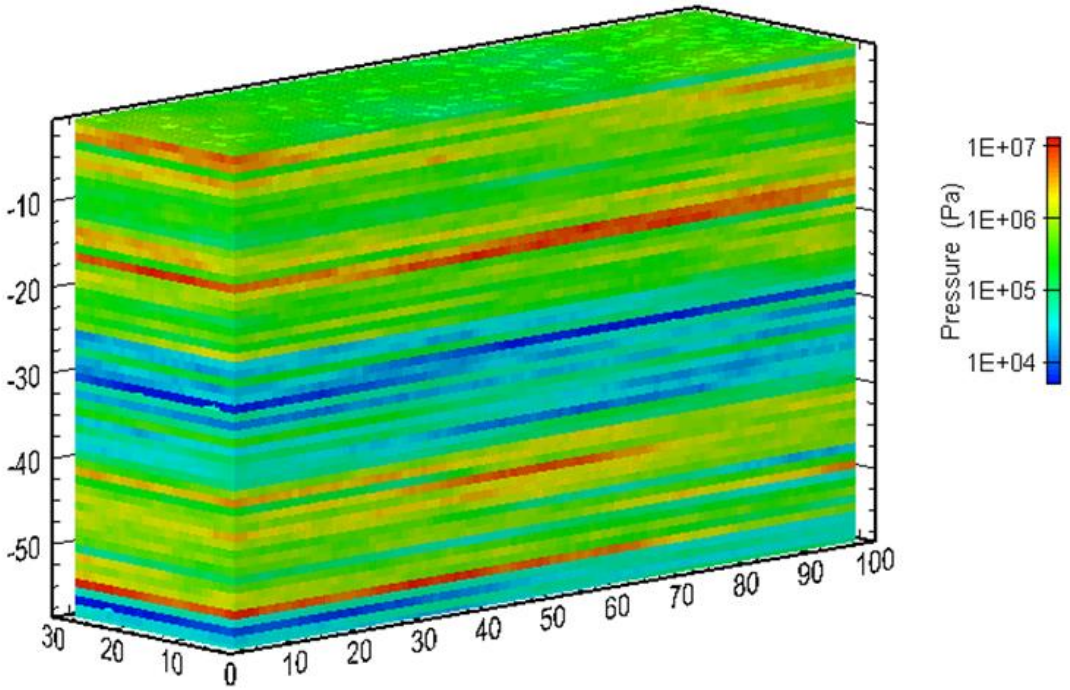



Figure 4. Random field distribution. An example random field input for variable spanning pressures. Higher pressures represent claystone and lower pressures represent sand.

The next program (RANDFIELD) transforms the random fields to the correct input format and values for the MMIP3 code. The model domain is discretized into an array of 1 $m^3$ grid blocks. Random fields of spanning pressure are generated by SGSIM and are normally distributed in log space and are mean zero. This program reads in the values**,** adds the means specified for clay or sand, and then transforms them out of log space. The executable and code structure must be run for each stratigraphic layer in the domain and the user must specify which random field structure is being transformed. These output files are then read into the percolation code. The quantity and dimensions of the additional zones must be defined in the MMIP3 input file.

The MMIP3 code is based on the macro modified invasion percolation method reported by *Glass et al*. (2001) and is used to simulate displacement of the gas phase by a wetting fluid. The code is a descendent of the modified invasion percolation code of *Holt et al.,* (2003) which was used to investigate the influence of centrifugal forces on unsaturated flow in laboratory centrifuge experiments. The model differs from standard invasion percolation models through the definition of macro-scale capillarity where individual pore throats and necks are not considered. Instead, a near pore-scale block is defined and characterized by a local threshold spanning pressure or local block-scale breakthrough pressure that represents the conditions of the sub-grid network (*Holt et al*., 2003). The model incorporates a 3D clustering algorithm, simultaneous invasion and reinvasion of water and air, hysteresis in water and air drainage curves, distributed porosities and drainage parameters, and gas phase compression and trapping. This code uses pressure formulations within and around the domain boundaries.

We apply a no flow boundary on all sides of the domain except for the top which has an atmospheric pressure of 1.01 x $10^5$ Pa (pascal) and overburden pressure of 2.975 x $10^5$ Pa for 20 meters of standing water calculated by

$$P_w = P_{atm} + \rho g h \tag{1}$$

where $P_w$ is the water overburden pressure (Pa), $\rho$ is the density of water assumed to be 1000 kg·m$^{-3}$, $g$ is gravitational acceleration with a value of 9.81 m·s$^{-2}$, and *h* is the depth of the water column. An invasion pressure for each block is then determined by summing the spanning pressures, buoyancy forces and viscous forces. The MIP model incorporates both capillary and buoyancy forces, and the invasion pressure is defined as

$$P_I = P_S + (\rho^d - \rho^i) g z \tag{2}$$

where $P_S$ is the capillary spanning pressure (ML$^{-1}$T$^{-1}$), $\rho^d$ is the density of the defending phase (ML$^{-3}$), $\rho^i$ is the density of the invading phase, $g$ is the acceleration due to gravity in the z-direction (LT$^{-2}$), and $z$ is the coordinate in the z direction (L). The term $(\rho^d - \rho^i) g z$ represents the buoyancy forces. Invasion pressures are determined by the pressure differences across the domain. The program locates all the active nodes in contact with the invading fluid. Those nodes are sorted by their invasion pressure and the node with the lowest invasion pressure is invaded first. An invasion percolation algorithm then sorts the invadable nodes, selects the node connected to the growing cluster with the lowest invasion pressure, and invades it. The process is repeated until the gas pressures in the cluster cells are large enough to inhibit further invasion.

A restart file is output every user defined invasion step from the percolation code and includes details about the entrapped gas clusters such as their cell index, pressures, and partial invasion factors. The diffusion restart program (DIFFRESTART) reads the domain dimensions and pressures from the restart file. We use sequential restart files where each file has greater clusters in the domain than the previous sequence, hence the invasion sequence continues after dissolution and diffusion. This successive starting and stopping of the percolation code represents the real kinetic growth event that would occur in nature, that is, a wetting front pushes air downward, the air is transported through the water to the atmosphere, and the wetting front continues to advance. The user must specify the restart file name in the open statement at the beginning of the program. The program first indexes the cluster pressures to their cell identification number. Next, the partial invasion factors need to be modified for fully occupied cells. The pressures for each cluster cell are multiplied by their corresponding partial invasion factor because the diffusion code assumes 1 $m^3$ grid blocks are fully and not partially occupied by gas. Multiplying by the partial invasion factors decreases the cell pressures so that every 1 $m^3$ cluster cell is fully occupied by gas. Then, the cluster cell pressures are averaged across the number of cells the cluster occupies. This pressure array represents the sub-grid domain beneath the overburden water pressure where zeros represent active cells (water) and initial condition array for DIFF_PERC. An array that lists every cluster number and the number of cells it occupies is output for averaging the mass flux out of a cluster in the diffusion program.

The change in concentration of a diffusing gas due to a concentration gradient is modeled by the transient diffusion equation. We consider transport by molecular diffusion without any flow through the system and neglect dispersive effects. Our model assumes that the transport of a dissolved gas in a porous medium can be described by Fick's second law, which states that an increase in the concentration in a cross section of unit area with time is the difference between the flux in to the volume and flux out of the volume. It is a parabolic equation derived from the fundamental laws describing the flux of concentrations using the law of conservation of mass. The three-dimensional diffusion equation is given as

$$\nabla \bullet \left( D \nabla C \right) = \frac{\partial C}{\partial t} \tag{3}$$

where $D$ is the effective diffusion coefficient, $C$ is concentration, and $t$ is time. A 3D backwards in time, block centered, integrated finite difference approach to approximate equation 3 for equidimensional grid blocks. We can represent equation 3 as a mass flux balance

$$\left\{ Mass\ flux\ in - Mass\ flux\ out \right\} = \left( C_{i,j,k}^{n} - C_{i,j,k}^{n-1} \right) \frac{\triangle x \triangle y \triangle z}{\triangle t} \tag{4}$$

where, subscripts $i$, $j$, and $k$ represent nodal locations (L), $n$ is a subscript indicating the time step (T), $C_{i,j,k}^{n}$ are unknown concentrations ($ML^{-3}$), and $C_{i,j,k}^{n-1}$ are concentrations from the previous time step.

When expanded equation 4 becomes

$$\frac{\triangle y \triangle z}{\triangle x}\left(J_{x_{in}} - J_{x_{out}}\right) + \frac{\triangle x \triangle z}{\triangle y}\left(J_{y_{in}} - J_{y_{out}}\right) + \frac{\triangle x \triangle y}{\triangle z}\left(J_{z_{in}} - J_{z_{out}}\right) = \left(C^{n}_{i,j,k} - C^{n-1}_{i,j,k}\right)\frac{\triangle x \triangle y \triangle z}{\triangle t} \qquad (5)$$

where $J_{x_{in}}$ and $J_{x_{out}}$ represent the mass flux in and out of a cell in the x direction ($ML^{-1}T^{-1}$), $J_{y_{in}}$ and $J_{y_{out}}$ represent the mass flux in and out of a cell in the y direction, $J_{z_{in}}$ and $J_{z_{out}}$ represent the mass flux in and out of a cell in the *z* direction, $\triangle t$ is the time step size, $\Delta x$ is the width of the cell in the x direction, $\triangle y$ is the width of the cell in the y direction, and $\triangle z$ is the width of the cell in the z direction. The flux terms in equation 5 can be defined as

$$\begin{aligned}
J_{x_{in}} &= J_{x_{i-1/2,j,k}} = -D_{i-1/2,j,k}\frac{C^{n}_{i,j,k} - C^{n-1}_{i-1,j,k}}{\triangle x} \\
J_{x_{out}} &= J_{x_{i+1/2,j,k}} = -D_{i+1/2,j,k}\frac{C^{n-1}_{i+1,j,k} - C^{n}_{i,j,k}}{\triangle x} \\
J_{y_{in}} &= J_{y_{i,j-1/2,k}} = -D_{i,j-1/2,k}\frac{C^{n}_{i,j,k} - C^{n-1}_{i,j-1,k}}{\triangle y} \\
J_{y_{out}} &= J_{y_{i,j+1/2,k}} = -D_{i,j+1/2,k}\frac{C^{n-1}_{i,j+1,k} - C^{n}_{i,j,k}}{\triangle y} \\
J_{z_{in}} &= J_{y_{i,j,k-1/2}} = -D_{i,j,k-1/2}\frac{C^{n}_{i,j,k} - C^{n-1}_{i,j,k-1}}{\triangle z} \\
J_{z_{out}} &= J_{y_{i,j,k+1/2}} = -D_{i,j,k+1/2}\frac{C^{n-1}_{i,j,k+1} - C^{n}_{i,j,k}}{\triangle z}
\end{aligned} \qquad (6)$$

where the effective diffusion coefficients *D* are defined as the harmonic means *i.e.*

$$D_{i,j,k+1/2} = \frac{2}{\frac{1}{D_{i,j,k}} + \frac{1}{D_{i,j,k+1}}} \qquad (7)$$

The integrated finite-difference model, where concentration is a function of space and time, is represented by incorporating the terms from equation 6 into equation 5 and results in

$$
\begin{aligned}
&-\frac{\triangle y\triangle z}{\triangle x}D_{i-1/2,j,k}(C^n_{i,j,k}-C^n_{i-1,j,k})-\frac{\triangle x\triangle z}{\triangle y}D_{i,j-1/2,k}(C^n_{i,j,k}-C^n_{i,j-1,k})-\frac{\triangle x\triangle y}{\triangle z}D_{i,j,k-1/2}(C^n_{i,j,k}-C^n_{i,j,k-1})-\\
&-\frac{\triangle y\triangle z}{\triangle x}D_{i+1/2,j,k}(C^n_{i,j,k}-C^n_{i+1,j,k})-\frac{\triangle x\triangle z}{\triangle y}D_{i,j+1/2,k}(C^n_{i,j,k}-C^n_{i,j+1,k})-\frac{\triangle x\triangle y}{\triangle z}D_{i,j,k+1/.2}(C^n_{i,j,k}-C^n_{i,j,k+1})\\
&=\left(C^n_{i,j,k}-C^{n-1}{}_{i,j,k}\right)\frac{\triangle x\triangle y\triangle z}{\triangle t}
\end{aligned}
\tag{8}
$$

Next, if we define the constants

$$
\begin{aligned}
D_1&=-\frac{\triangle y\triangle z}{\triangle x}D_{i-1/2,j,k}\quad D_2=-\frac{\triangle x\triangle z}{\triangle y}D_{i,j-1/2,k}\quad D_3=-\frac{\triangle x\triangle y}{\triangle z}D_{i,j,k-1/2}\\
D_4&=-\frac{\triangle y\triangle z}{\triangle x}D_{i+1/2,j,k}\quad D_5=-\frac{\triangle x\triangle z}{\triangle y}D_{i,j+1/2,k}\quad D_6=-\frac{\triangle x\triangle y}{\triangle z}D_{i,j,k+1/2}
\end{aligned}
\tag{9}
$$

and

$$
E=\frac{\Delta x\Delta y\Delta z}{\Delta t}
\tag{10}
$$

equation 8 becomes

$$
\begin{aligned}
&D_1(C^n_{i,j,k}-C^n_{i-1,j,k})+D_2(C^n_{i,j,k}-C^n_{i,j-1,k})+D_3(C^n_{i,j,k}-C^n_{i,j,k-1})-\\
&D_4(C^n_{i,j,k}-C^n_{i+1,j,k})+D_5(C^n_{i,j,k}-C^n_{i,j+1,k})+D_6(C^n_{i,j,k}-C^n_{i,j,k+1})=\left(C^n_{i,j,k}-C^{n-1}{}_{i,j,k}\right)E
\end{aligned}
\tag{11}
$$

Collecting like terms from equation 11 and multiplying both sides by -1 results in

$$
\begin{aligned}
&-D_1(C^n_{i-1,j,k})-D_2(C^n_{i,j-1,k})-D_3(C^n_{i,j,k-1})+C^n_{i,j,k}(D_1+D_2+D_3+D_4+D_5+D_6+E)-\\
&-D_4(C^n_{i+1,j,k})-D_5(C^n_{i,j+1,k})-D_6(C^n_{i,j,k+1})+C^n{}_{i,j,k}E=C^{n-1}{}_{i,j,k}E
\end{aligned}
\tag{12}
$$

Equation 12 is an equation for node *i, j, k.* Deriving an equation for all $n = nxnynz$ nodes results in a set of *n* simultaneous equations being solved for *n* unknown concentration values. Equation 12 can be re-written as

$$\mathbf{A} \bullet \mathbf{c} = \mathbf{f} \tag{13}$$

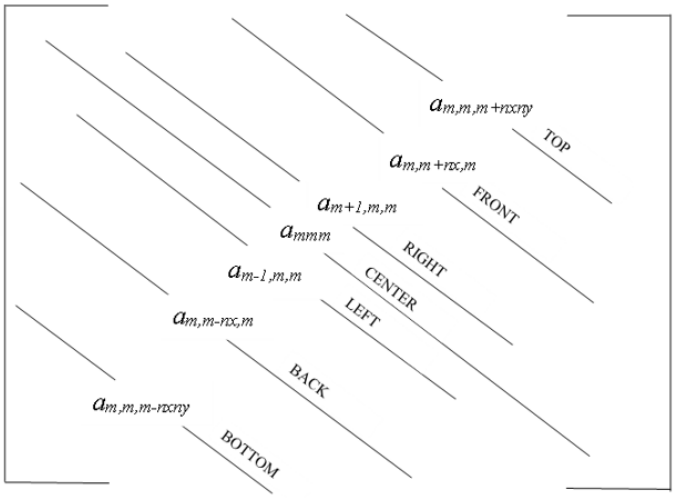
and its inverse as

$$\mathbf{c} = \mathbf{A}^{-1} \bullet \mathbf{f} \tag{14}$$

where $\mathbf{c}$ is the vector of unknown concentrations, $\mathbf{A}$ is the coefficient matrix that contains the values on the left hand side of equation 12, and $\mathbf{f}$ is the load vector that contains the values on the right hand side of equation 12. The coefficient matrix is a symmetric, $mxm$ sparse banded matrix whose values depend on $\Delta x$ , $\Delta y$ , $\Delta z$ , and *D.*

The matrix is a 7 point grid and defined as

A=




(15)

where *m* is a global indexing scheme defined as

$$m = (k-1) \cdot nx \cdot ny + (j-1) \cdot ny + i \tag{16}$$

so that

$$C_{i,j,k} = C_m \tag{17}$$

The main diagonal (center) $a_{mmm}$ includes values from the left hand side of equation 12 and is defined as

$$a_{mmm} = (D_1 + D_2 + D_3 + D_4 + D_5 + D_6 + E) \tag{18}$$

The load vector $\mathbf{f}$ contains information about boundary conditions and source and sink terms. We automatically fill these locations with zeros. The modification to the coefficient matrix set up is later discussed. For boundary conditions, the load vector and coefficient matrix need to be modified by adding a defined constant to each side. For a constant concentration boundary on the top,

$$J_{in} = -2D\frac{\Delta x \Delta y}{\Delta z}(C^n_{i,j,k} - \mathrm{bvT}) \tag{19}$$

where bvT is the top boundary value. The coefficient matrix is modified by

$$a_{mmm} = a_{mmm_{old}} + 2D\frac{\Delta x \Delta y}{\Delta z} \tag{20}$$

and the load vector by

$$f_m = f_{m_{old}} + 2D\frac{\Delta x \Delta y}{\Delta z} bvT \tag{21}$$

to reflect the boundary conditions. The new concentrations are then solved as before at time $t^n$.

The model was verified by comparing its results to analytical solutions for simple cases of molecular diffusion. We solved the numerical model for a non-steady state diffusion in a one-dimensional (1D) plane sheet with uniform initial distributions and equal surface concentrations as well as heat flow in a two-dimensional (2D) finite rectangle with faces having unit initial temperatures. The 1D and 2D models were subject to the following initial concentrations and boundary conditions

$$C(x,0) = 1.0$$
$$C(x,y,0) = 1.0$$
$$C = 0 \text{ for } x = \pm l$$
$$C = 0 \text{ for y} = \pm b$$

The 1D solution in the form of a trigonometrical series is given by *Crank* (1975) as

$$C = C_0 \frac{4}{\pi} \sum_{n=0}^{N} \left[ \frac{1}{2n+1} \sin\left[ \frac{2n+1\pi x}{l} \right] \exp\left[ -\frac{(2n+1)^2 \pi^2 Dt}{l^2} \right] \right] \quad (22)$$

where $C_0$ is the initial concentration, $l$ is distance from the middle, $x$ is the x coordinate location, $D$ is the effective diffusion coefficient, and $t$ is time. We set $x$ equal to 58 which results in a $l$ value of $\frac{x}{2}$. The normalized percent error between the numerical results and 1D analytical solution was less than 0.003% (Figure 5).

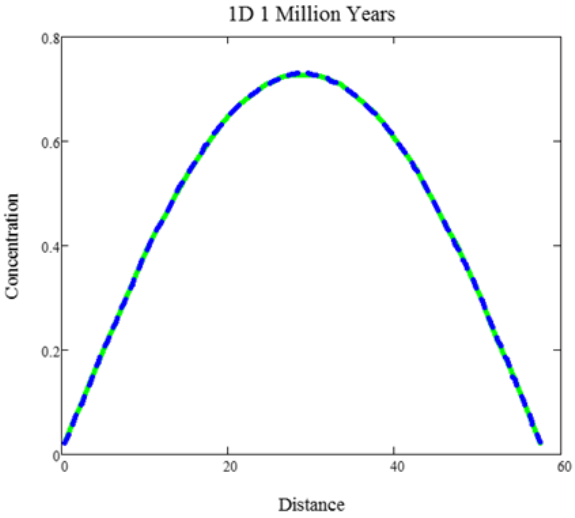



Figure 5. 1D verification. 1D analytical solution (green) from *Crank* (1975) equation 4.16 compared to numerical results (blue) of the diffusion code for an initial concentration of 1 over the entire domain.

The 2D solution of *Carslaw and Jaegar* (1959) is given as

$$\psi(x,l)=\frac{4}{\pi}\sum_{n=0}^{N}\left[\frac{-1^{n}}{2n+1}\exp\left[\frac{-D(2n+1)^{2}\pi^{2}t}{4l^{2}}\right]\cos\left[\frac{\left[(2n+1)\pi x\right]}{2l}\right]\right]$$
$$\psi(y,b)=\frac{4}{\pi}\sum_{n=0}^{N}\left[\frac{-1^{n}}{2n+1}\exp\left[\frac{-D(2n+1)^{2}\pi^{2}t}{4l^{2}}\right]\cos\left[\frac{\left[(2n+1)\pi y\right]}{2l}\right]\right] \quad (23)$$

$$C=\psi(x,l)\psi(y,b) \quad (24)$$

where $\psi$ denotes the concentration with respect to *x, y, l,* and *b*. We set *l* equal to 98 and *b* equal to 28 to match our model dimensions. The normalized percent error between the numerical results and 2D analytical solution were less than 0.5% (Figure 6).

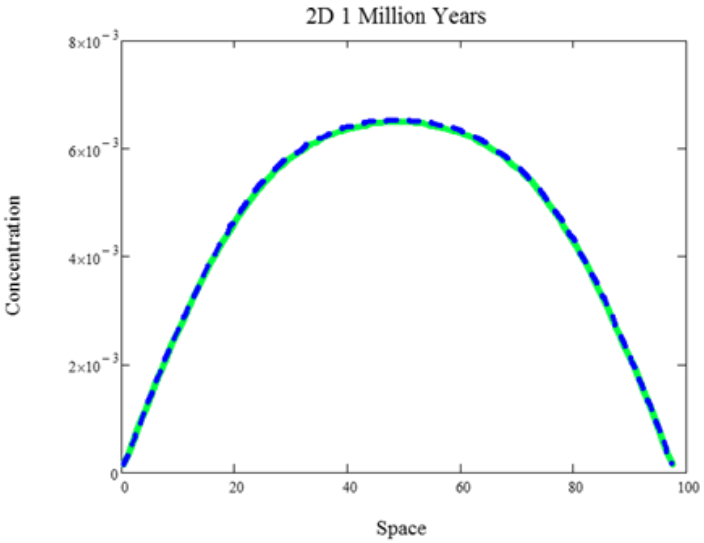



Figure 6. 2D verification. 2D analytical solution (green) from *Carslaw and Jaegar* (1959) compared to numerical results (blue) of the diffusion code for an initial concentration of 1.

Furthermore, the principle of conservation of mass requires that the net flux (*i.e.* the cumulative sum of mass inflows and outflows) must equal the accumulation of mass over time. The corresponding mass of a diffusing substance remaining after a given amount of time was compared to the analytical solution of *Crank* (1975) which is

$$M = \sum_{n=0}^{N}\left[\frac{8}{(2n+1)^2\pi^2}\exp\left[\frac{-D(2n+1)^2\pi^2 t}{4l^2}\right]\right] \tag{25}$$

where, $M$ is the mass remaining in the system after a given time $t$. The normalized percent error between the numerical results and the mass analytical solution were less than 0.1% (Figure 7).

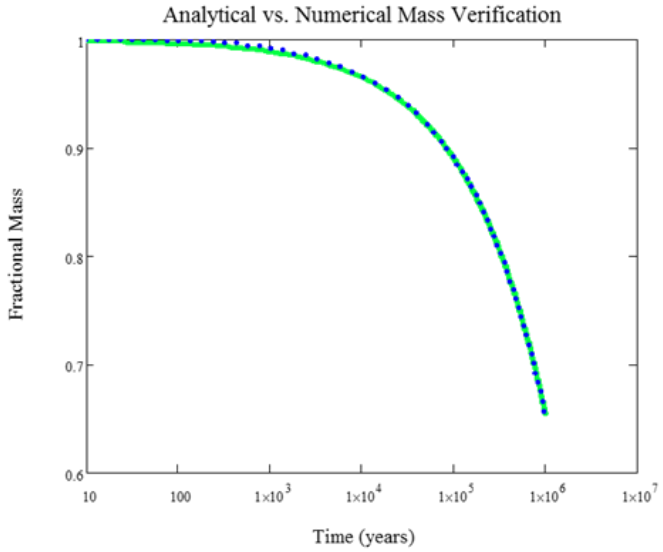



Figure 7. Mass verification. Numerical results from the diffusion code (blue) and analytical solution (green) from *Crank* (1975) equation 4.18. Approximately 65% of the air mass remains after 1 MY with an effective diffusion coefficient= $10^{-11}$m$^2$s$^{-1}$.

Normalized mass balance errors were calculated by

$$M_b = \frac{M_f - \Delta M_s}{M_o - \Delta M_{cs}} \tag{26}$$

where $M_f$ is the net mass flux out of the system, $\Delta M_s$ is the change in mass of the system, $M_o$ is the initial mass in the domain, and $\Delta M_{cs}$ is the cumulative change in mass out of the system. The normalized values of $M_b$ at the end of test simulations were approximately ±1 %.

The diffusion code is incorporated into the MMIP3 model by using the trapped air cluster properties from the restart file. The diffusion code structure, at a high level, describing the principal segments can be seen in Figure 8. Detailed user instructions and more intensive explanation of the programs subroutines can be found Appendix A.

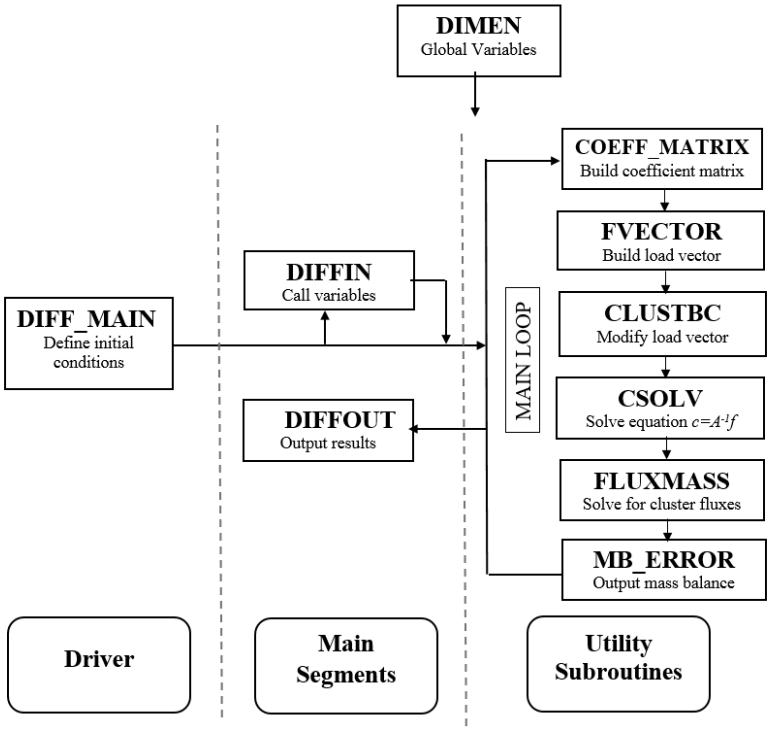



Figure 8. DIFF_PERC flow chart. Sequence of subroutines and loops for the diffusion program.

A dimension module (DIMEN) declares and sets the global variables used for all of the subroutines. It is used to reduce excessive variable declarations for each subroutine individually. The module contains specifications and definitions that can be used by one or more program subroutines. For the module to be accessible, the other program subroutines must reference its name.

The main input subroutine (DIFFIN) is required to run the diffusion code and reads information from the main input file. Input variables and initial conditions defined may be found in Appendix A. Nitrogen comprises approximately 78% of the earth atmosphere, therefore, we assume the gas has the same physical properties. The pressures from the MMIP3 code are converted to concentrations to accurately model diffusive transport.

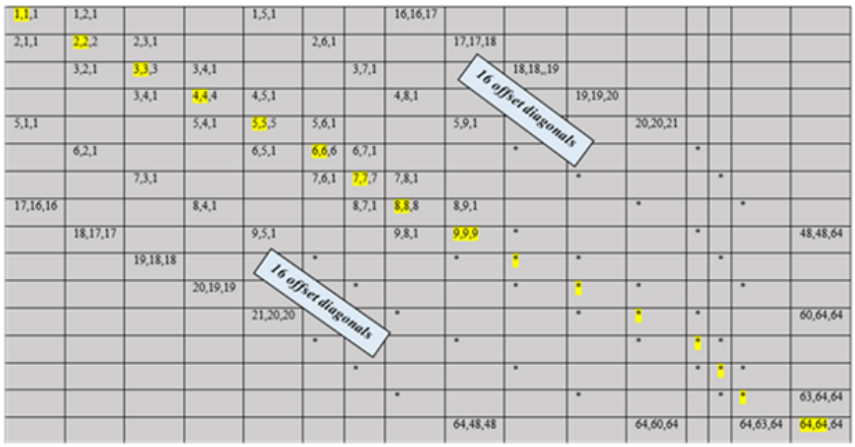
The initial gas cluster pressures are adjusted to concentrations using the equation

$$C_i = \frac{C_i^{gas}}{K_{H,i}(T,S)} = \frac{p_i}{R \cdot T \cdot K_{H,i}(T,S)} \tag{27}$$

from *Holocher et al.*, 2002, where $C_i$ is the equilibrium concentration of the gas $i$, $p_i$ is the partial pressure in the gas phase, $R$ is the universal gas constant, and $K_{H,i}$ is the dimensionless Henry coefficient which depends on the temperature $T$ and salinity $S$. A $K_{H,i}$ value of 63.5 was used for $N_2$ at 20° Celsius (*Holocher et al.*, 2002). We set the solubility concentration in the wetting phase surrounding the clusters equal to $\approx$20 mg/L. The terms 'partial pressure' and concentration may be used proportionally as they are correlated to the ideal gas law.

The coefficient matrix subroutine (COEFF_MATRIX) sets up the domain in a band storage structure. The band storage structure of the coefficient matrix saves both storage, space, and computation time. A banded matrix is stored so the *j*th column of the matrix corresponds to the $j^{th}$ column of the array. This iterative technique requires less storage than direct methods for sparce $mxm$ matrices since the zero diagonals between the main diagonal and the outermost non-zero diagonals do not need to be stored. The storage is a $mxn$ band matrix with three lower non-zero sub-diagonals (left, front, top) and three upper non-zero super-diagonals (right, back, bottom) representing the cell direction from the center nodes in the main diagonal. They are stored in a compact two-dimensional array with $nxny+1$ rows and $nxnynz$ columns. Columns of the matrix are stored in the corresponding columns of the array and diagonals of the matrix are stored in the corresponding rows of the array (Figure 9).

i.)




ii.)

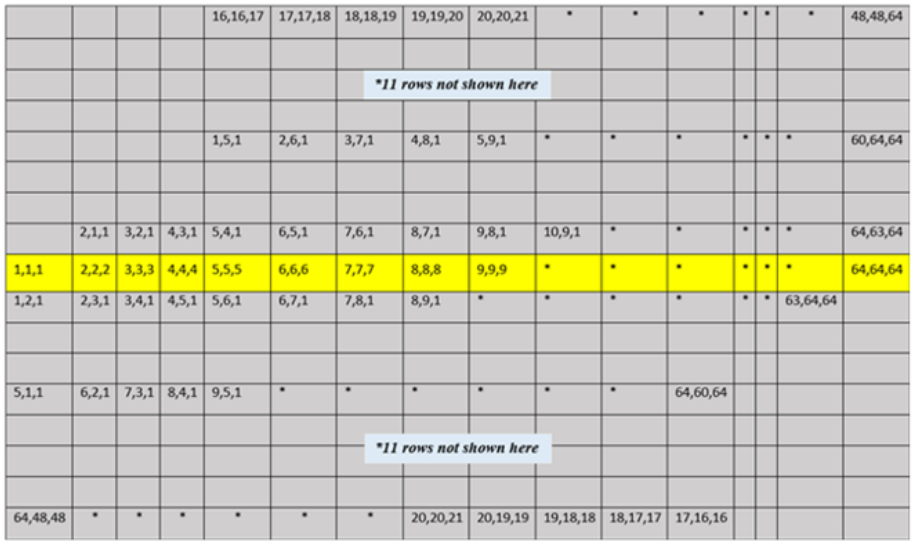



Figure 9. Coefficient matrices. A regular *mxm* sparse matrix (i) and *mxn* 3D coefficient matrix in band storage (ii) that forms *2nxny+1* rows and *nxnynz* columns with 3 upper and 3 lower co-diagonals (rows). Stars represent values not listed to save space and gray spaces are zeros. The yellow highlighted row is the main diagonal of the matrix.

The rows and columns in matrix **A** are built to reflect the conditions for all sides of the domain, including the main diagonal. The offset between the rows in band storage reflect the same number of zero diagonals that would occur in a regular sparse *mxm* matrix. This technique essentially rotates the diagonal matrix to reduce the number of zero elements in a smaller storage spectrum.

The cells that occupy a cluster are not included in the coefficient matrix and are not solved for in the system of equations. The length of the coefficient matrix needs to be decreased by the number of cluster cells to reflect these internal boundaries.

The load vector subroutine (FVECT) sets up the load vector $\mathbf{f}$ of the system of equations and modifies the coefficient matrix $\mathbf{A}$ for the specified boundary conditions. The main diagonal of the matrix is first calculated followed by the other domain sides. The main diagonal represents the right hand side of equation 12 and is calculated for every node. The rate of change in concentration at a point in space is proportional to the second derivative of concentration with space. The load vector and coefficient are modified to reflect the boundary types prescribed. Detailed calculations can be seen in Appendix A.

A cluster cell is a region of continuous cells that have the defending phase (gas) in them. They represent the volume and mass of the defending phase that opposes invasion. Concentrations across a cluster are distributed evenly and are assumed to have the same concentrations irrespective of the cell elevation. The cluster boundary subroutine (CLUSTBC) locates active cells bounding cluster cells and identifies which boundary sides of those cells are adjacent to a cluster cell. Cluster cells completely bounded by other cluster cells *i.e.* the center of a cluster are not indexed. An indexing scheme is first defined for all the active cells adjacent to a cluster cell for the every side of a cell. If a cell in the cluster index array has a value of 1 then it is a cluster cell and 0 if it is an active cell. The first portion of the subroutine traverses across the domain and locates the cells with a value of 1 (cluster cell) and defines the indexed boundary conditions for that cell.

These values will be used in calculating the net mass flux out of an entire cluster in the flux mass subroutine. All cluster cells on the boundary nodes of the domain are accounted for next. If a cluster cell is located on the left edge boundary of the domain then it cannot have a boundary condition cell to the left of it. This rule is applied to the other domain boundaries. If a cluster cell is bounded by another cluster cell on any side then an indexed boundary condition is not defined for that side of the cell (Figure 10).

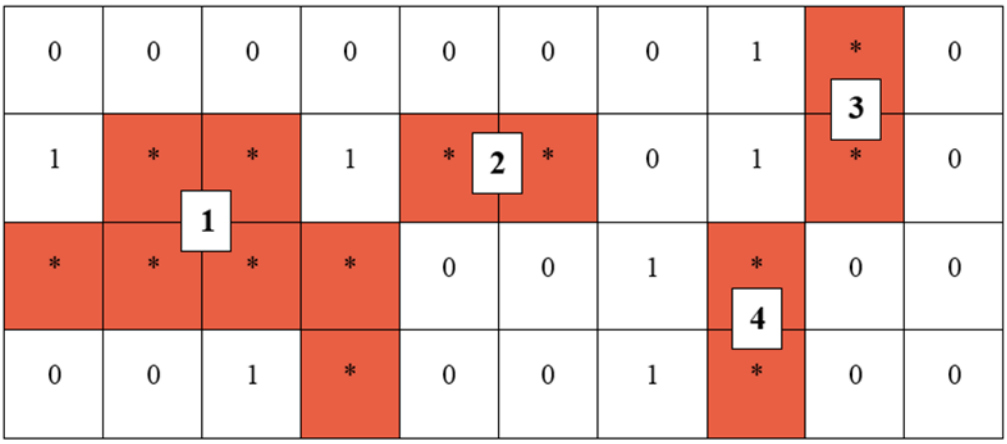



Figure 10. Cluster indexing scheme. Example indexing scheme for defining cluster boundary conditions to the left of a cluster cell**.** There are 4 clusters here (red). A 1 is defined for a cluster cell that has an adjacent cell to the left. The other sides are accounted for in the same indexing loop.

A cluster cell surrounded by cluster cells on all 6 sides will have a value of 0 defined for the boundary cell index condition for all sides. Finally, the load vector $\mathbf{f}$ and coefficient matrix $\mathbf{A}$ are modified for those active cells that have boundary conditioned values. If an active cell is adjacent to a cluster cell then the coefficient matrix and load vector must be updated. We treat the cluster cells as internal boundaries, therefore, we update the coefficient matrix and load vector to reflect these boundaries before the subroutine ends.

The solver subroutine (CSOLV) solves the linear equation $\mathbf{A} \bullet \mathbf{c} = \mathbf{f}$ using a preconditioned conjugate gradient method (*e.g. Hageman and Young*, 1981), where $\mathbf{A}$ is the band storage coefficient matrix, $\mathbf{f}$ is the load vector, and $\mathbf{c}$ are the unknown concentrations being solved for. The preconditioning technique transforms the matrix equation and a generalized conjugate gradient method (*e.g. Golub and Van Loan*, 1983) is applied to the transformed matrix. IMSL subroutines are required (*IMSL FNL 7.0.1*). The IMSL subroutine LFCQS factors the matrix and checks for non-positive definiteness or ill-conditions. The PCGRC subroutine solves a real symmetric definite linear system using the preconditioned conjugate gradient method by utilizing two other IMSL operations. The first is MURBV which multiplies the real band matrix in band storage mode $\mathbf{A}$ by the real vector $\mathbf{f}$. Then, LSLQS solves the real symmetric definite system of linear equations $\mathbf{c} = \mathbf{A}^{-1} \bullet \mathbf{f}$ in band storage mode without iterative refinement. Lastly, the newly calculated concentrations are cycled through the rest of the program in the next time step. The last portion of this subroutine outputs the concentrations per a user defined time step size for post processing visualization purposes.

The cluster boundary indexing scheme is applied in the next subroutine (FLUXMASS) to locate cluster cells bounded by an active (wetted) cell for all directions and calculate the mass flux out of each cluster. The flux is the change in concentration across the cluster cell and its adjacent bounded active cell for all sides (Figure 11). The sum of these mass fluxes are then subtracted from their original (previous) mass per volume of the cluster. These cluster concentrations reflect the change in mass per cluster and are defined as the new cluster concentrations for the next time step. An example of the mass flux derivation can be found in Appendix A.

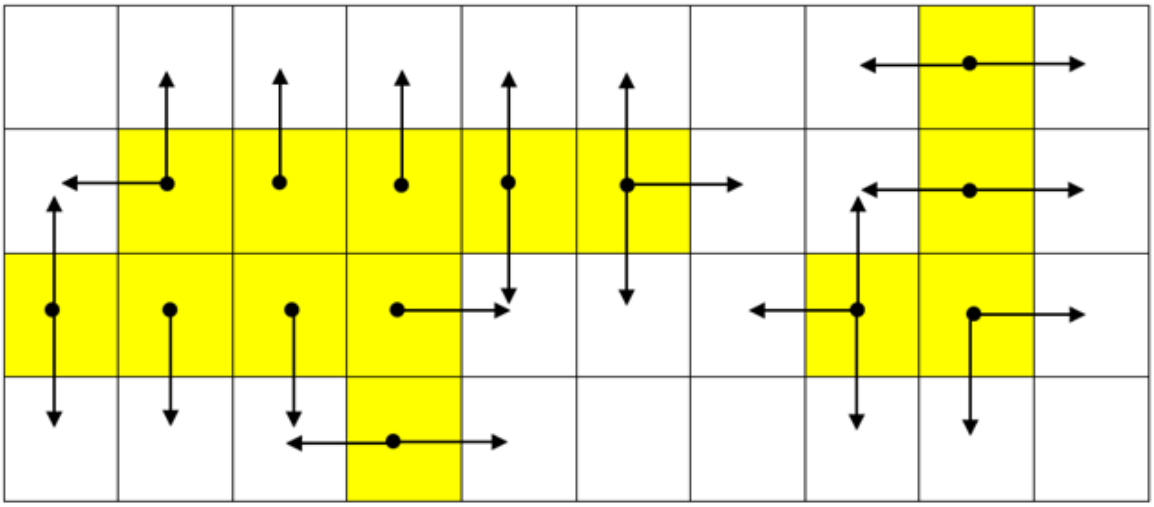


Figure 11. Cluster flux conceptual model. The total mass flux out of these two clusters (yellow) is the sum of the fluxes (arrows) out of every cluster cell adjacent to an active cell (white). This figure does not include the adjacent upper and lower layers.

Mass balance errors are output every time step in the subroutine (MBERROR). The principle of the conservation of mass requires that the net mass flux per time step must equal the net change in mass within the domain during that time step. The change in flux into the domain with the change in flux out of the domain should have a small relative error. The fluxes reflect the boundary value type and are the concentration difference across the boundary of an active cell and the boundary condition value (*e.g.*, constant concentration or no flow). The mass balance error will be negative when there is more gas exiting across the boundaries than is lost from the total domain each time step. First, the max flux out of each domain boundary is calculated. The cumulative total flux is calculated by adding the mass flux from the previous time step with the newly calculated flux. The sum of mass in the domain is subtracted from the initial total mass in the domain (total mass in the domain from the previous time step) to reflect the quantity of mass leaving the domain.

Other mass calculations include a mass residual error, a cumulative mass balance error, a cumulative normalized mass balance error, the cumulative mass out, a cumulative mass residual error, the fractional total mass out of the domain, the fractional mass (residual) left in the domain, and a cumulative time calculation. The last portion of this subroutine updates mass variables and defines the size of the next time step**.**

To reduce computational run time for a lengthy scenario (*e.g.*, 1 MY), the initial time step is multiplied by a time scale factor until a defined maximum time step is reached, then each time step is set to the maximum time step size for the rest of the program. Detailed calculations can be found in Appendix A.

After the main loop of the program is complete, the final array of concentrations is output in DIFFOUT**.** This subroutine has the option to output the grid center location of each node in the *x*, *y*, and *z* directions. The decreased cluster cell concentrations are inserted back in to their original indexed cell location *m* defined at the beginning of the program and the other active cells are output to their corresponding cell number in the global indexing scheme. The concentrations are converted back to pressures using the inverse of equation 27. The output file is the main input file for reconstructing the restart file used in the percolation code.

The main driver of the program (DIFFMAIN) calls the subroutines previously discussed, redefines initial conditions, and allocates all the arrays. It is an organized sequence of call statements to the primary subroutines. After the main loop of the program and final output of pressures, the arrays are deallocated to release the allocated memory and the program is ended.

After the diffusion code has run, a separate program (DIFF_REPERC) reads in the pressures from the diffusion code file and imports them back in to the restart file used for the MMIP3 code. An inactive top layer of nodes needs to be appended to the file to match the original domain size of the initial restart file. The user must specify which restart file is being updated in the open statement at the beginning of the program. Next, the program indexes the cluster pressures from the diffusion code with their coinciding cell location and averages them with their partial invasion factor and number of cells (cell volume) occupied. For clusters that occupy only one cell, the pressures are divided by their original partial invasion factor to 'squeeze' the pressures in to their original partial volume occupied. For clusters larger than one cell, the pressures are averaged across the cluster volume occupied. The output file is a mirror image of the initial restart file except for the newly calculated and reduced cluster pressures. It is used as the initial restart input file for the MMIP3 code and must be user defined in the MMIP3 input file.

The percolation code and diffusion code are coupled until the gas phase equilibrates with the wetting phase (invasion stops). The MMIP3 model is restarted a final time to simulate the expansion of trapped air following the removal of the standing water, using the reduced pressures. The remaining trapped air is allowed to expand (if diffusive losses are minimal) and equilibrate with the atmosphere. Simulation results can be compared to pressures and volume of compressed air present in low-permeability soils. The impact of variations in the approximate duration of the MIP process and the duration of the diffusion process can be systematically evaluated. The primary focus is on the mass of trapped air remaining, local- and model-scale saturation, and local capillary pressure profiles. Differences in saturation and other hydrologic parameters can then be analyzed and compared that reflect the effects of diffusion on the system.

# III. RESULTS

The model domain is discretized into a three dimensional cartesian grid, with *nx*, *ny*, and *nz* grid blocks in the *x*, *y*, and *z* directions (Figure 12). Each grid block is assumed to have the same dimension widths *dx*, *dy*, *dz* in the *x*, *y*, and *z* directions, respectively. The indices. *i*, *j*, and *k* are used to denote grid block numbers in the *x*, *y*, and *z* directions. The grid blocks at the two ends in each direction are considered inactive in the percolation code, so the dimensions used in the diffusion code reflect the active sub-grid domain in the percolation code. Therefore, we used sub-grid dimensions of 98 *x* 28 *x* 78 for a 100 *x* 30 *x* 60 domain with 20 meters of water on top. We apply no flow boundaries on all sides except for the top that has a constant atmospheric concentration of $\approx$20 mg/L.

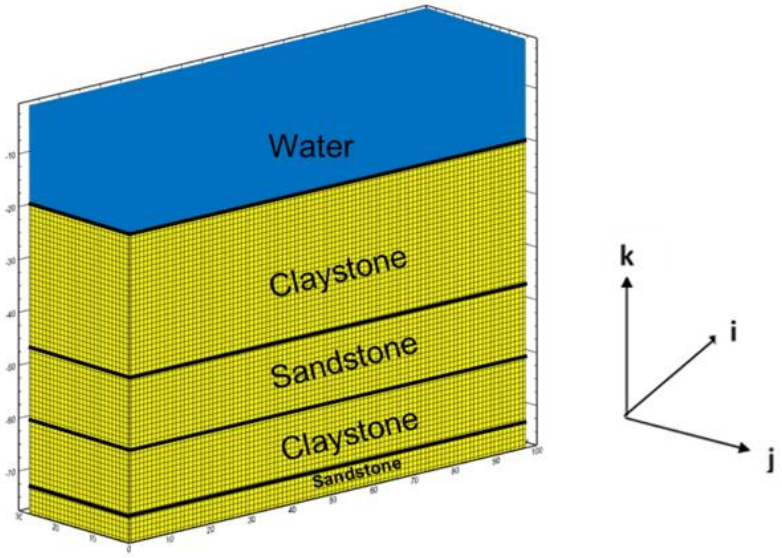



Figure 12. Discretization of the model domain. The top 25 meters is claystone underlain by a 15 meter thick sandstone and claystone unit followed by a 5 meter sandstone unit at the bottom.

As an example application, field data was acquired from the Waste Control Specialists (WCS) site in west Texas where radioactive waste is currently being housed in a sequence of low-permeability mudrocks. The geology at the site has been examined through cores and excavations (as well as by cuttings and geophysical logs) to provide a local framework for characterizing the hydrogeologic properties of the rocks (*Holt et al*., 2008). For the low-permeability mudrock, we assume an effective diffusion coefficient of $10^{-11} m^2 s^{-1}$ (*Schloemer*, 2004). The main variables obtained from field data were the covariance structure of the soil types from discontinuity mapping reports (*Holt et al*., 2011) with geometric mean values of 2.2 and 1.8 MPa (log means of 8.65 and 5.95), and log variance values of 2.7 and 1.1 for clay and sand, respectively. We used a free-water diffusion coefficient of $10^{-9} m^2 s^{-1}$ (*Appelo and Postma*, 2010).

We ran a sequence of eight simulations, resulting in a total elapsed time of 10,000 years. Each of the first seven diffusion simulations were 1,000 years and the last compilation was 3,000 years. The wetting phase in the percolation code advanced after each coupling sequence with the diffusion code. The wetting starts out on the top of the domain. Due to strong capillary forces in the clay, the wetting phase completely percolated through the 25 meter clay layer. The water preferably invaded the rest of the clay layer first due to higher capillary forces before percolating through the sand layer. After the clay layer was invaded, the water moved in to the sand layer (Figure 13).

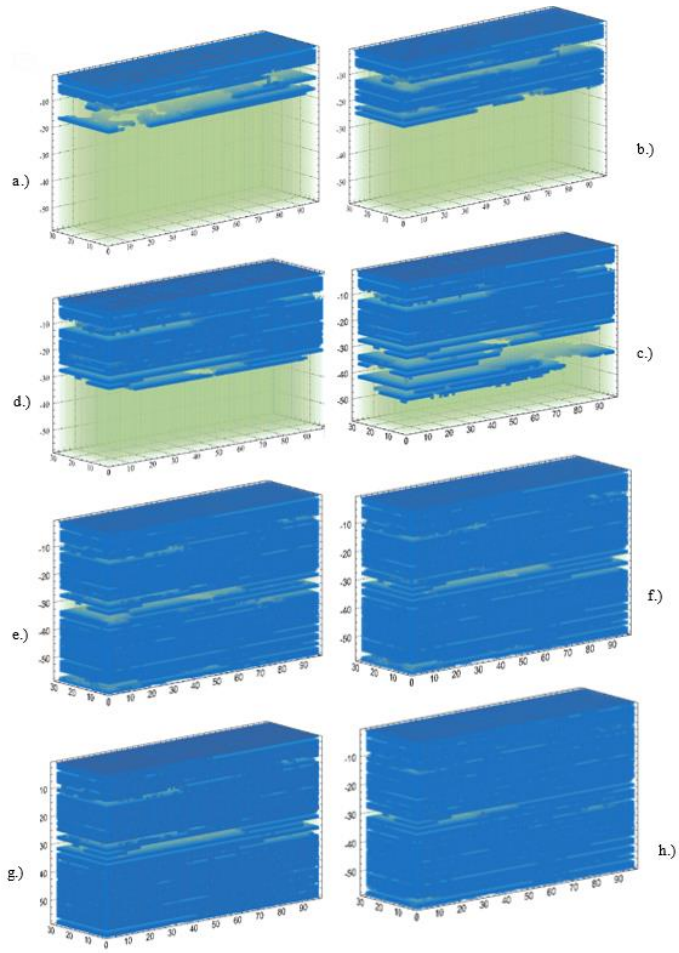



Figure 13. Percolation code wetting phase. Invasion sequence (a-h) of the wetting phase every ten thousand invasion steps. A correlated sand layer exists at 25 meters (c).

The capillary forces in sand are lower, so gravity had a larger contribution in invasion into the sand layer. Therefore, the water tended to move vertically through the sand. As water increasingly invaded the sand, the large volume of trapped air kept shrinking and splitting into independent clusters. This occurred until the trapped gas shrunk to small enough pockets that the internal pressures were high enough to resist further invasion. The pressure in the majority of clusters were more than six to eight times atmospheric pressure. The pressures were mapped as initial concentrations for the diffusion code (Figure 14).

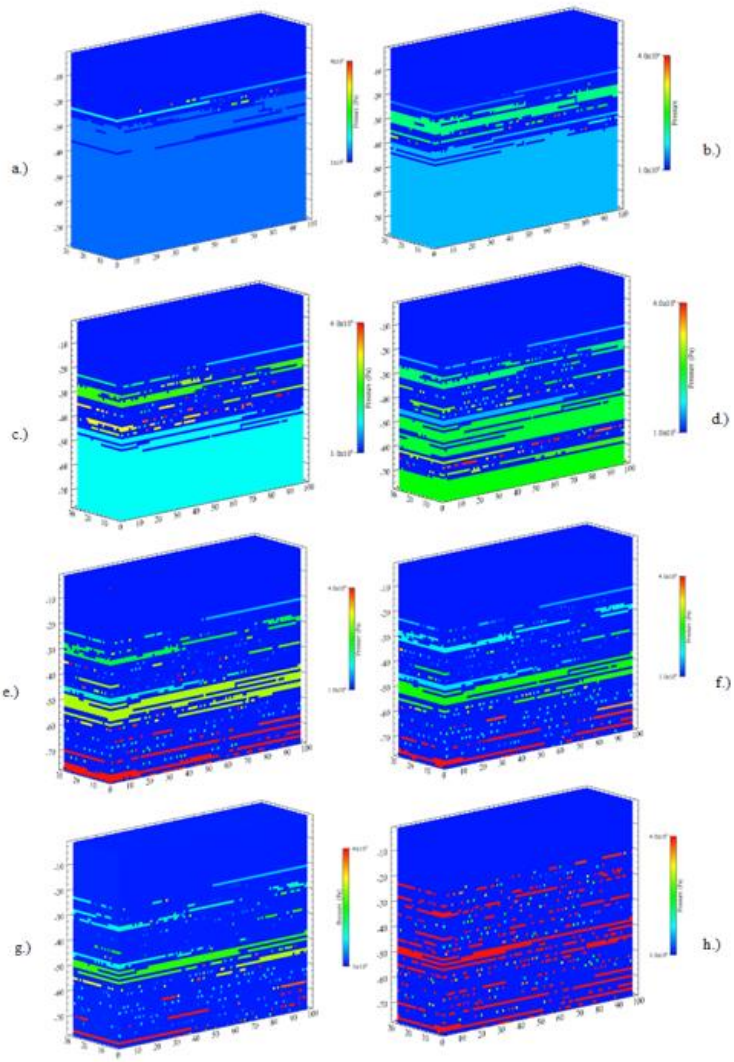



Figure 14. Initial concentration distributions. Pressures from the percolation code were mapped as initial concentrations (a-h) in the diffusion code for every simulation sequence.

The pressure of residual air at the beginning of each diffusion simulation progressively increased as the wetting phase advanced in the percolation program. This confirms the idea that as a wetting front advances, trapped air becomes compressed and the gas pressures in the clusters increase (Figure 15).

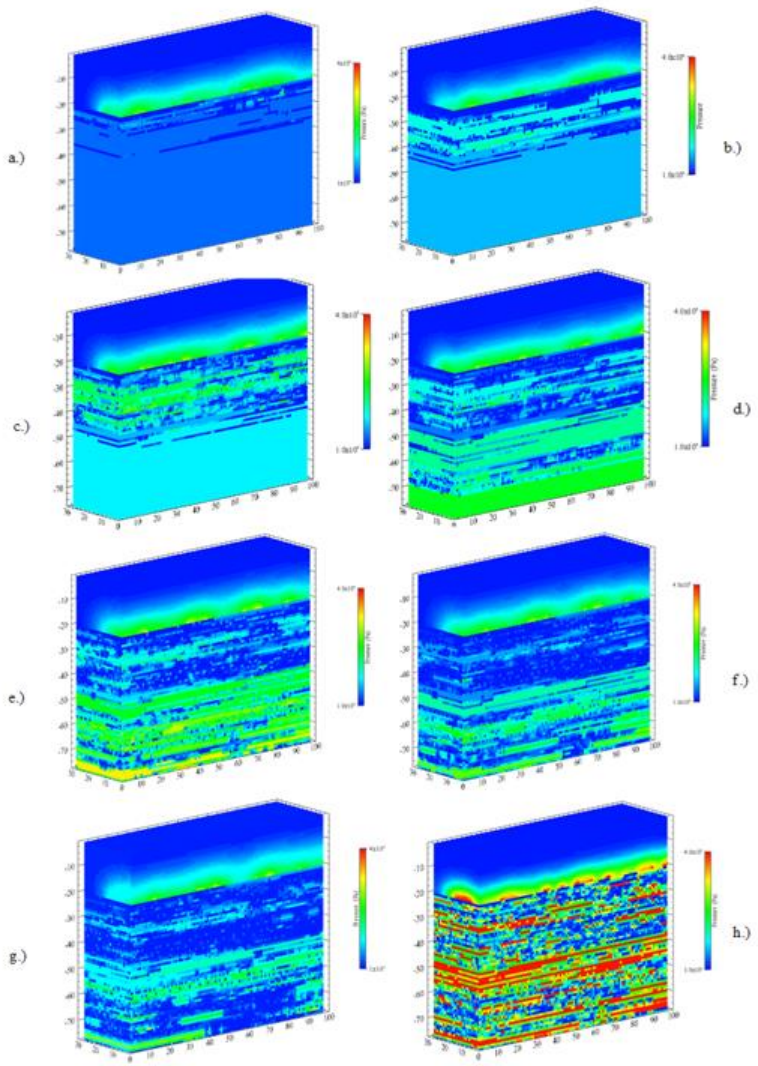



Figure 15. Final concentration distributions. A total diffusion sequence of ten thousand years where (a-g) represent the end of one thousand years and h.) represents the final concentrations after three thousand years.

The percolation code accounts for simultaneous reinvasion by water and air after removing the overburden pressure. As the standing water overburden above the top clay unit was removed, and the domain becomes connected to the atmosphere, we found that most of the entrapped gas clusters became disconnected by water reinvasion rather than expanding in each invasion sequence (Figure 16).

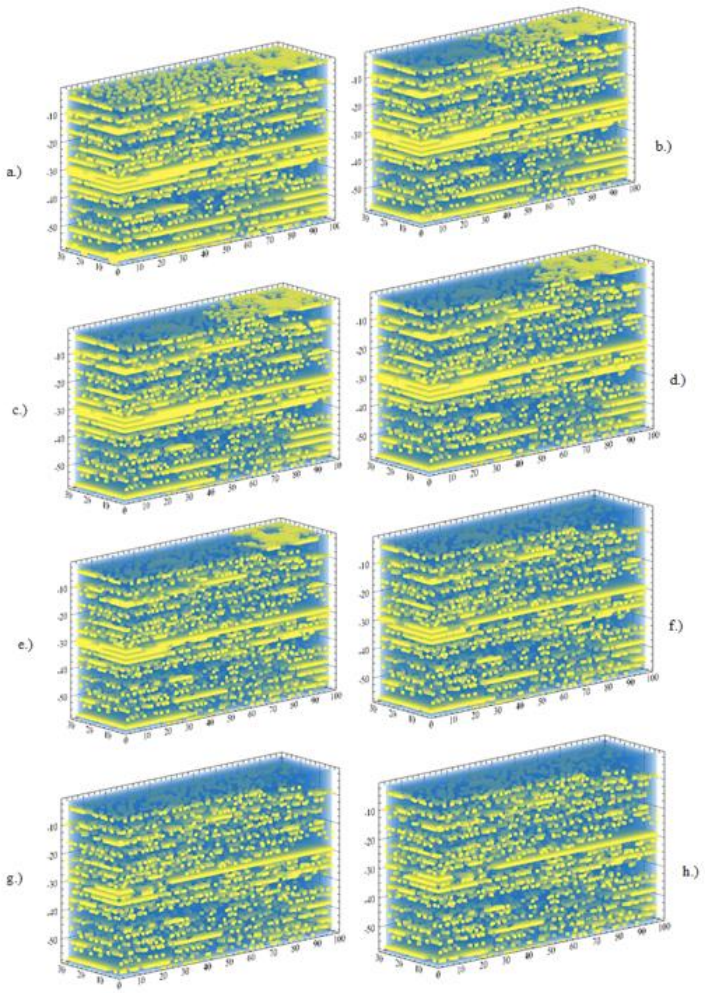



Figure 16. Overburden removal sequence. Reinvasion of water and air (a-h) every 1000 time steps after the overburden pressure is removed. The number of air clusters (yellow) double from reinvasion by water and the average air pressure decreases.

This decrease in expansion and splitting of continuous air clusters due to invasion of pores by water indicates that dissolution decreased the cluster pressures and their ability to expand.

Numerous simulations were attempted. A few of the sequenced simulations were successful in coupling the diffusion code with the percolations code, using the reduced cluster pressures, but there were discrepancies and limitations when altering a restart file by reducing the cluster pressures for the percolation code. A successful coupling was determined by cluster pressures in the restart files. We found that running the diffusion code for a time greater than 1 to 3,000 years drops the cluster pressures low enough ($\approx 0.2$ MPa) that the percolation code produced error messages. If a cluster contains a single cell and is disconnected from the atmosphere, the cell can instantaneously equilibrate (disappear) with the atmosphere when the water is removed. The cluster cell is given a negative pressure and the other cluster indices are not updated to reflect this. This caused a discrepancy when using the percolation code and the updated restart file because it didn't correlate with the indexing scheme applied. This occurrence was very common in the sequence attempts, and therefore the number of simulations presented were fewer than desired.

Simulation results varied. We simulated a continuous invasion sequence and expansion of trapped air after equilibrium without intermittent restart files and diffusive losses versus diffusion for 10,000 years. For an invasion percolation sequence without diffusion and reduced cluster pressures, the saturation (assuming the domain is initially air) and average air pressure were 0.86 and 0.78 MPa, respectively. Diffusive losses resulted in an increased saturation of 0.98 and decreased average air pressure of 0.68 MPa (5% decrease). The number of clusters of size 1 nearly doubled from 3,344 to 6,074. When dissolution occurred by applying the transport mechanisms of diffusion, the saturation in the domain increased $\approx 1\%$ after each 1,000 year diffusion simulation. Although the quantity of clusters increased, the decreased average air pressure resulted in more water reinvasion (splitting) into the larger and contiguous clusters.

When the average air pressures in the clusters are higher, the probability of water reinvasion splitting larger continuous clusters decreases.

For further analysis, we decreased the effective diffusion coefficient by three orders of magnitude to $10^{-14}$ $m^2s^{-1}$, simulated a continuous invasion sequence, and modeled diffusion for 10,000 years. The rate of change in concentration of the clusters decreased dramatically (Figure 17). The average air saturation increased only 0.001% after removing the water and the average air pressure decreased by 0.002 %.

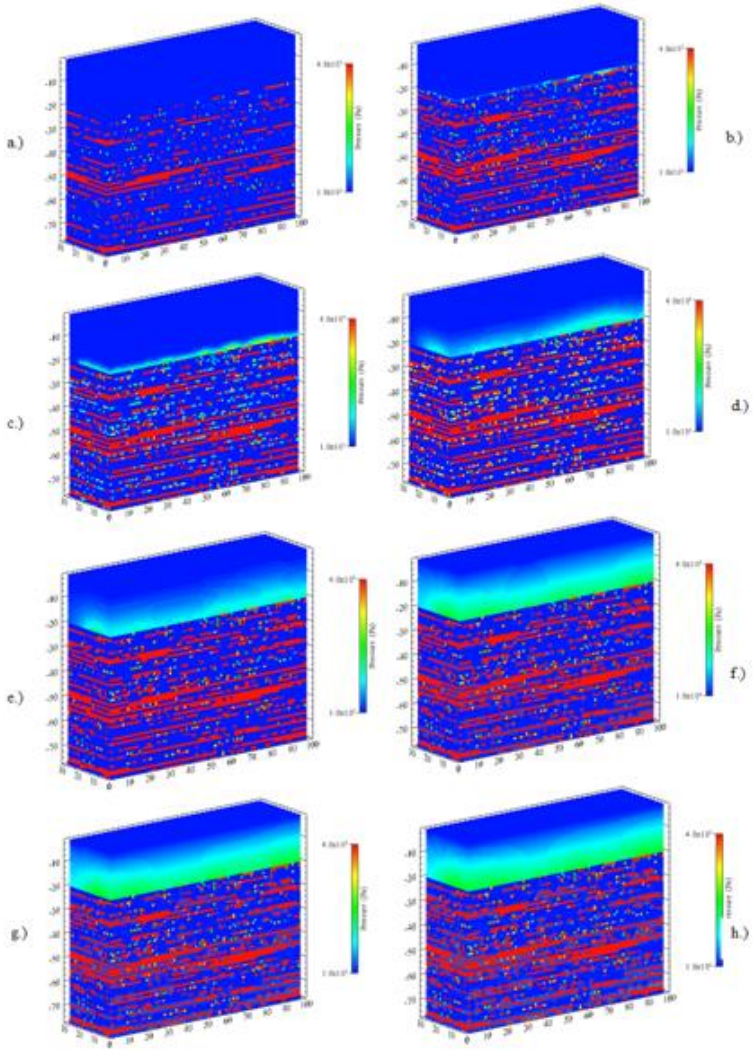



Figure 17. Alternate diffusion coefficient sequence. Ten thousand year simulation (a-h) using an effective diffusion coefficient of $10^{-14}$ $m^2$.

# IV. SUMMARY AND DISCUSSION

We presented a FORTRAN 90 code that can be successfully coupled with an invasion percolation code to simulate the evolution, dissolution, and expansion of trapped and compressed gases in low permeability soils. A backwards in time integrated finite-difference approach was applied to solve for unknown concentrations. The diffusion code was verified in 1D and 2D systems with analytical solutions analogous to heat flow. Spatially correlated spanning pressures with a log normal distribution were produced for the percolation code. We successively started and stopped the diffusion code 8 times to simulate diffusion for 10,000 years. Average air pressures and saturation values were compared before and after incorporating the diffusion code. There were obvious discrepancies and limitations in the percolation code when the cluster pressures were altered (decreased) in the restart files. The effective diffusion coefficient of $10^{-11}$ $m^2/s$ used may actually be lower in nature due to microbial activity, adsorption and desorption, temperature, salinity, and variable diffusive flux characteristics( *e.g.*, local concentrations of a diffusing species may be retarded by gas partitioning between the mobile aqueous phase and a stationary trapped gas phase with time as diffusion proceeds) (*Fry*, 1995).

Given the current state of the percolation code, successful coupling with the diffusion code depends on the modeled length of time desired. In general, as the cluster pressures decrease, uncertainty arises in successfully running the MMIP3 code.

For future investigations, we suggest directly incorporating the diffusion code in to the percolation code structure itself or investigating why the decreased cluster pressures produce errors when restarting the percolation model. An outside program may be written which would run the diffusion program after every invasion step in the percolation code. This would essentially simulate simultaneous invasion percolation with diffusion and discrepancies of the lengthy and extensive percolation program would not need to be addressed. Future simulations may be conducted to evaluate long periods of subaerial exposure of a low permeability surface and the behavior of local capillary pressures in response to boreholes open to the atmosphere. The ability to apply the complex phase simulations proposed here can aid in determining whether or not a geologic unit is suitable for housing hazardous or radioactive waste. This research also enhances the understanding of environmental issues associated with the diffusion of compressed and trapped air through a porous medium and can be applied for other scenarios, such as $CO_2$ leakage from injection boreholes at sequestration sites.

LIST OF REFERENCES

# V. REFERENCES


Adrian, D.D., Franzini, J.B., 1966, Impedance to infiltration by pressure build-up ahead of the wetting front, Journal of Geophysical Research, v. 71, p. 5857-5862.

Aeschbach-Hertig, W., Beyerle, U., Holocher, J., Peeters, Kipfer, R., 2001, Excess air in groundwater as a potential indicator of past environmental changes, Water Resources and Drinking Water, p. 174-183.

Appelo, C.A.J., and Postma, D., 2010, Geochemistry, groundwater and pollution, 2nd Edition, p. 88.

Bloomsburg, G.L., and Corey, A.T., 1964, Diffusion of entrapped air from porous media, Hydrology Papers, Colorado State Univ., Fort Collins, CO. No. 5.

Bodman, G.B., 1937, The variability of the permeability "constant" at low hydraulic gradients during saturated water flow in soils, Soil Science Society of America Journal, v. 2, p.45-53.

Carslaw, H. S., & Jaeger, J. J. C., 1959, Conduction of Heat in Solids, Oxford Science Publications, p. 173.

Crank, J., 1975, The Mathematics of Diffusion, Oxford Science Publications, p. 44-62.

Deutsch, C.V., Journael, A.G., 1998, GSLIB: Geostatistical software library and users guide. New York, Oxford University Press.

Dixon, R.M., and Linden, D.R., 1972, Soil air pressure and water infiltration under border irrigation, Soil Science Society of America, Proceedings, v. 33, p. 948-953.

Donaldson, J.H., Istok, J.D., OReilly, K.T., 1998, Dissolved Gas Transport in the Presence of a Trapped Gas Phase: Experimental Evaluation of a Two- Dimensional Kinetic Model, Groundwater, v. 36, p. 133-142.

Fry, V.A., Istok, J.D., Semprini, L., O'Reilly, K.T., Buscheck, T.E., 1995, Retardation of dissolved oxygen due to a trapped gas phase in porous media, Groundwater, v. 33, p. 391-398.

Glass, R. J., Rajaram, Nicholl, M.J., Detwiler, R.L., 2001, The interaction of two fluid phases in fractured media, Current Opinion in Colloid & Interface Science, v. 6, p. 223-235.

Glass, R. J., Nicholl, M.J., Rajaram, H., Andre B., 2004, Development of slender transport pathways in unsaturated fractured rock: Simulation with modified invasion percolation: Geophysical Research Letters, v. 31, p. 1944-8007.

Golub, G.H., and Van Loan, C.F., 1983, Martrix Computations 4th Edition, p. 625.

Hageman, L.A., and Young, D.M., 1981, Applied iterative methods, Academic Press, New York, Chapter 7.

Holocher, J., Peeters, F., Aeschbach-Hertig, W., Hofer, M., Brennwald, M., Kinzelbach, W., and Kipfer, R., 2002, Experimental investigations on the formation of excess air in quasi-saturated porous media, Geochimica et Cosmochimica Acta, v. 66, p. 4103-4117.

Holocher, J., Peeters, F., Aeschbach-Hertig, W., Kinzelbach, W., Kipfer, R., 2003, Kinetic Model of Gas Bubble Dissolution in Groundwater and its Implications for the Dissolved Gas Composition, Environmental Science & Technology, v. 37, p. 1337-1343.

Holt, R. M., Glass, R. J., Sigda, J. M., Mattson, E. D., 2003, Influence of centrifugal forces on phase structure in partially saturated media, Geophysical Research Letters, v. 30, NO. 13, 1692.

Holt, R.M., Hughes, E., and Hubbell, J.M., 2008, In situ hydrogeologic conditions determined from core samples collected in the vicinity of the Federal Waste Landfill and a playa north of the Federal Waste Landfill at the Waste Control Specialists Site, Andrews County, Texas: Prepared for Waste Control Specialists LLC, P.O. Box 1129, Andrews, Texas.

Holt, R.M., Grisak, G.E., Pickens, J.F., Powers, D.W., Kuszmaul, J., Hughes, E.E., Griffith, C., Cook, S.L., 2010, Conceptual Model Report: submitted as an attachment to a letter from William P. Dornsife, WCS, to Susan Jablonski, Texas Commission on Environmental Quality, on June 24, 2010.

Holt, R.M., J.S. Kuszmaul, D.W. Powers, and E.E. Hughes, 2011, Subsurface Discontinuity Mapping for the Federal Waste Disposal Facility and Compact Waste Disposal Facility Landfills," Submitted as an attachment to a letter from William P. Dornsife, WCS, to Susan Jablonski, Texas Commission on Environmental Quality, on September 26, 2011.

IMSL Fortran Numerical Library (FNL) Version 7.0.1 for Intel ® 64, Microsoft Windows 64-bit

Ioannidis, M.A., Chatzis, I., and Dullien, A.L., 1996, Macroscopic percolation model of immiscible displacement: Effects of buoyancy and spatial structure: Water Resources Research, v. 32, p. 3297-3310.

Ioannou, I., Hall, C., Wilson, M.A., Hoff, W.D., and Carter, M.A., 2003, Direct measurement of the wetting front capillary pressure in a clay brick ceramic, Journal of Physics D: Applied Physics, v. 36, p. 3176-3182.

Jalali-Farahani, H.R., Heermann, D.F., and Duke, H.R., 1993, Physics of surge irrigation, II, Relationship between soil physical and hydraulic parameters, Transactions of the American Society of Agricultural Engineers, v. 36, p. 37-44.

Klump, S., Cirpka, O.A., Surbeck, H., Kipfer, R., 2008, Experimental and numerical studies on excess-air formation in quasi-saturated porous media, Water Resources Research, v. 44, p. 1-15.

Kueper, B.H., and McWhorter, D.B., 1992, The use of macroscopic percolation theory to construct large-scale capillary pressure curves: Water Resources Research, v. 28, p. 2425-2436.

Latifi, H., Prasad., S.N., and Helweg., O.J., 1994, Air entrapment and water infiltration in two-layered soil column, Journal of Irrigation and Drainage Engineering, v. 120, p. 871-879.

McWhorter, D.B., 1971, Infiltration affected by flow of air, Hydraulic Properties of Porous Media, Paper. 49.

Morel-Seytoux, H.J., and Khanji, J., 1974, Derivation of an equation of infiltration, Water Resources Research, v.10, p. 795-800.

Morel-Seytoux, H.J., and Billica, J.A., 1985, A two-phase numerical model for prediction of infiltration: Application to semi-infinite soil column, Water Resources Research, v. 21, p. 607-615.

Navarro, V., Yustres, A., Candel, B., and Garcia, B., 2008, Soil air compression in clays during flood irrigation, European Journal of Soil Science, v. 59, p. 799-806.
Parlange, J.-Y., Hill, D.E., 1976, Theoretical analysis of wetting front instability in soils, Soil Science, v. 122, p. 236-239.
Peck, A.J., 1965, Moisture profile development and air compression during water uptake by air-confining porous bodies, Soil Science, v. 100, p.44-51.

Rolston, D.E., 1986, Gas diffusivity, Methods of soil analysis, 2nd edition, p. 1089-1102.
Sander, G.C., Parlange, J.-Y., and Hogarth, W.L., 1988, Air and water flow, II, Gravitational flow with an arbitrary flux boundary condition, Journal of Hydrology, v. 99, p. 225-234.
Schloemer, S., Krooss, B.M., 2004, Molecular transport of methane, ethane and nitrogen and the influence of diffusion on the chemical and isotopic composition of natural gas accumulations, Geofluids, v.4, p. 81-108
Singh, A., Clemo, T., Ramarao, B.S., 2010, MMIP3_Air_WC: A Code for Macro Modified Invasion Percolation: Prepared for Waste Control Specialists LLC, P.O. Box 1129, Andrews, Texas.
Toboada, M.A., Lavado, R.S., Rubio, G., Cosentino, D.J., Soil volumetric changes in natric soils cause by air entrapment following seasonal ponding and water table rises, Geoderma, v. 101, p. 49-64.
Wang, Z., and Feyen, J., Van Genuchten, M.T., Nielsen, D.R., 1998, Air entrapment effects on infiltration rate and flow instability, Water Resources Research, v. 32, p. 213-222.
Wilkinson, D., and Willemsen, J.F., 1983, Invasion percolation: A new form of percolation theory: Journal of Physics A: Mathematical and General, v. 16, p. 3365-3376.
Wilson, L.G., and Luthin, J.N., 1963, Effects of air flow ahead of the wetting front on infiltration, Soil Science, v. 91, p. 137-143.

LIST OF APPENDICES

# APPENDIX A: Diffusion Code Manual

TABLE OF CONTENTS

## 1.0 BACKGROUND

A new 3D diffusion code (DIFF_PERC) has been developed and coupled with an MMIP3 (3D Modified Macroscopic Invasion Percolation) model to allow for diffusive losses of trapped and compressed air (gas) to be quantified. Trapped gas can be created in an unconfined aquifer due to water level fluctuations caused by seasonal changes in aquifer recharge or discharge (*Fry*, 1995). The soil air, initially at atmospheric pressure, can become compressed ahead of the wetting front during recharge events following flooding (*Navarro et al*., 2008). Considering drainage of a porous medium, air enters from the top and enters large pores first. Small pores remain water filled and air continues to move into large pores connected to the top. Air invasion stops when the air pressure is not sufficient to invade additional pores. Entrapped air may also occur by water moving in to an originally dry soil. Advancing wetting fronts push air downward and trap air in large pores (Figure A-1). As the water pressure increases, zones of trapped air get compressed, leaving zones of interconnected compressed air at depth. A complicated history of wetting and surface drying often leads to these conditions (*Holt et al.*, 2010). For low permeability soils, capillary pressures in core can average 3 MPa while the capillary pressures in instrumented boreholes average 0.3 MPa (*Holt et al*., 2008). These discrepancies suggest the presence of a trapped and compressed gas phase. Trapped gas in a porous medium has been shown to be removed by diffusion (*Bloomsburg and Corey*, 1964). When gas is trapped in a bubble form, the gas will diffuse out of the bubble into the surrounding aqueous phase and then diffuse to the surface of the water table where the aqueous concentration is in equilibrium with the atmospheric pressure (*Fry*, 1995) (Figure A-2). An analytical solution for diffusion of the air phase indicates that trapped air can persist for millions of years in low

permeable soils (*Crank,* 1975) (Figure A-2). Given the long time scale for the problem, it is assumed that dissolution of trapped air is limited by diffusion, not the kinetics of dissolution.

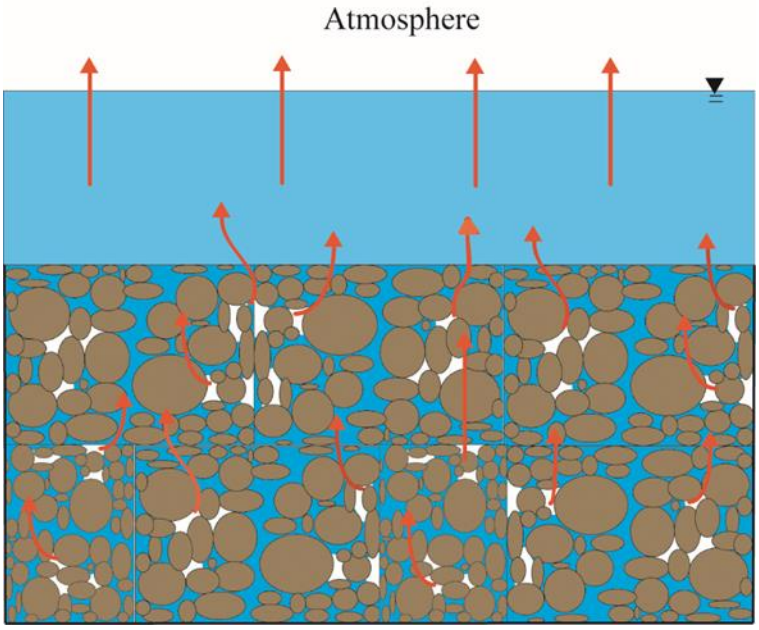



Figure A-1. Entrapped air model. Conceptual model of entrapped air (white) in a partially saturated porous media with overburden of standing water (blue) and the diffusion process (red) to the atmosphere.

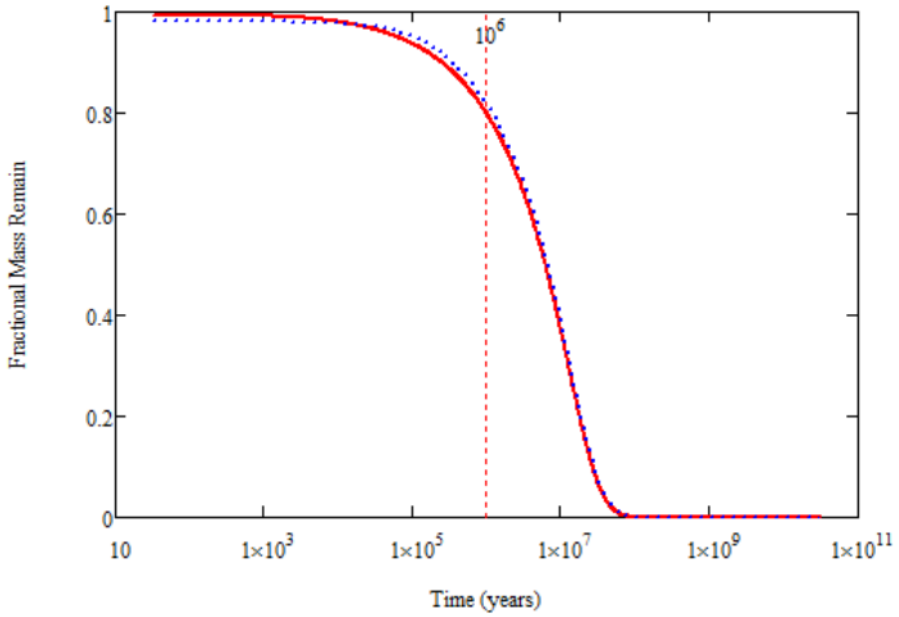



Figure A-2. Analytical mass solution. Approximately 80% of the initial mass remains after 1 MY with a diffusion coefficient of $10^{-11}m^2s^{-1}$ (Crank, 1975) equation 4.18.

A backwards in time, block-centered, integrated finite-difference approach is used to solve the diffusion equation on a uniform grid. The model is capable of supporting two diffusion coefficients for water and air and 1st and 2nd type boundary conditions. The model has been be verified by comparing numerical and analytical solutions for similar scenarios of molecular diffusion and analogous transport parameters (*e.g.*, heat transfer). The MMIP3 model is used to simulate the evolution of trapped and compressed gases following the transgression of water over a geologic unit. In baseline simulations, the MMIP model is normally allowed to evolve to a steady-state condition, but we halt the percolation code after a shortened wetting cycle. The pressure associated with each trapped zone is then mapped as initial air concentrations for the diffusion simulations. Diffusive losses from the trapped and compressed zones are then simulated using the diffusion code DIFF_PERC. The diffusion occurs for a user specified time and the pressure loss in the trapped zones can be determined. The MMIP3 code is then restarted from its previous state and stopped again. This successive starting and stopping of the percolation code represents the kinetic growth event that would occur in the field, that is, a wetting front pushes air downward, the air diffuses into the aqueous solution, and then the wetting front continues etc. Once equilibrium is attained in the wetting phase, the MMIP3 model is restarted a final time to simulate the expansion of trapped air following the removal of water, using the reduced pressures.

The objective of this code is to model the origin and fate of compressed and trapped gases within a low-permeability soil following the transgression and regression of water, specifically by the role of diffusion. The existing 3D MMIP model is coupled with the 3D diffusion code to accomplish this objective. The terms ‘pressure’ and concentration may be used interchangeably as they are correlated by the perfect ideal gas law. The pressures from the

MMIP3 code are converted to concentrations for the diffusion code then converted back to pressures for the percolation code. Other subprograms were written to couple and interface the diffusion code with the MMIP3 percolation code. Their construct and capabilities will be discussed in this manual.

### *1.1 Theory and Verification*

The change in concentration of a diffusing gas due to a concentration gradient is modeled by the transient diffusion equation. Our model assumes that the transport of a dissolved gas in a porous medium can be described by Fick's second law, which states that an increase in the concentration in a cross section of unit area with time is the difference between the flux in to the volume and flux out of the volume. It is a parabolic equation derived from the fundamental laws describing the flux of concentrations using the law of conservation of mass. The rate of change in concentration at a point in space is proportional to the second derivative of concentration with space *x, y,* or *z*. The three-dimensional diffusion equation is given as

$$\nabla \bullet \left( D \nabla C \right) = \frac{\partial C}{\partial t} \tag{1}$$

where *D* is the effective diffusion coefficient, *C* is concentration, and *t* is time. A 3D backwards in time, block centered, integrated finite difference approach to approximate equation 3 for equidimensional grid blocks. We can represent equation 3 as a mass flux balance

$$\left\{ Mass\ flux\ in - Mass\ flux\ out \right\} = \left( C_{i,j,k}^{n} - C_{i,j,k}^{n-1} \right) \frac{\triangle x \triangle y \triangle z}{\triangle t} \tag{2}$$

where, subscripts *i*, *j*, and *k* represent nodal locations (L), *n* is a subscript indicating the time step (T), $C^{n}_{i,j,k}$ are unknown concentrations ($ML^{-3}$), and $C^{n-1}_{i,j,k}$ are concentrations from the previous time step. When expanded equation 2 becomes

$$\frac{\triangle y \triangle z}{\triangle x}\left(J_{x_{in}} - J_{x_{out}}\right) + \frac{\triangle x \triangle z}{\triangle y}\left(J_{y_{in}} - J_{y_{out}}\right) + \frac{\triangle x \triangle y}{\triangle z}\left(J_{z_{in}} - J_{z_{out}}\right) = \left(C^{n}_{i,j,k} - C^{n-1}_{i,j,k}\right)\frac{\triangle x \triangle y \triangle z}{\triangle t} \tag{3}$$

where $J_{x_{in}}$ and $J_{x_{out}}$ represent the mass flux in and out of a cell in the x direction ($ML^{-1}T^{-1}$), $J_{y_{in}}$ and $J_{y_{out}}$ represent the mass flux in and out of a cell in the y direction, $J_{z_{in}}$ and $J_{zout}$ represent the mass flux in and out of a cell in the *z* direction, $\triangle t$ is the time step size, $\triangle x$ is the width of the cell in the x direction, $\triangle y$ is the width of the cell in the y direction, and $\triangle z$ is the width of the cell in the z direction. The flux terms in equation 3 can be defined as

$$\begin{aligned}
J_{x_{in}} &= J_{x_{i-1/2,j,k}} = -D_{i-1/2,j,k}\frac{C^{n}_{i,j,k} - C^{n-1}_{i-1,j,k}}{\triangle x} \\
J_{x_{out}} &= J_{x_{i+1/2,j,k}} = -D_{i+1/2,j,k}\frac{C^{n-1}_{i+1,j,k} - C^{n}_{i,j,k}}{\triangle x} \\
J_{y_{in}} &= J_{y_{i,j-1/2,k}} = -D_{i,j-1/2,k}\frac{C^{n}_{i,j,k} - C^{n-1}_{i,j-1,k}}{\triangle y} \\
J_{y_{out}} &= J_{y_{i,j+1/2,k}} = -D_{i,j+1/2,k}\frac{C^{n-1}_{i,j+1,k} - C^{n}_{i,j,k}}{\triangle y} \\
J_{z_{in}} &= J_{y_{i,j,k-1/2}} = -D_{i,j,k-1/2}\frac{C^{n}_{i,j,k} - C^{n-1}_{i,j,k-1}}{\triangle z} \\
J_{z_{out}} &= J_{y_{i,j,k+1/2}} = -D_{i,j,k+1/2}\frac{C^{n-1}_{i,j,k+1} - C^{n}_{i,j,k}}{\triangle z}
\end{aligned} \tag{4}$$

where the effective diffusion coefficients *D* are defined as the harmonic means *i.e.*

$$D_{i,j,k+1/2} = \frac{2}{\frac{1}{D_{i,j,k}} + \frac{1}{D_{i,j,k+1}}} \tag{5}$$

The integrated finite difference model, where concentration is a function of space and time, is represented by incorporating the terms from equation 4 into equation 3 and results in

$$\begin{aligned} &-\frac{\triangle y \triangle z}{\triangle x} D_{i-1/2,j,k}(C^n_{i,j,k} - C^n_{i-1,j,k}) - \frac{\triangle x \triangle z}{\triangle y} D_{i,j-1/2,k}(C^n_{i,j,k} - C^n_{i,j-1,k}) - \frac{\triangle x \triangle y}{\triangle z} D_{i,j,k-1/2}(C^n_{i,j,k} - C^n_{i,j,k-1}) - \\ &-\frac{\triangle y \triangle z}{\triangle x} D_{i+1/2,j,k}(C^n_{i,j,k} - C^n_{i+1,j,k}) - \frac{\triangle x \triangle z}{\triangle y} D_{i,j+1/2,k}(C^n_{i,j,k} - C^n_{i,j+1,k}) - \frac{\triangle x \triangle y}{\triangle z} D_{i,j,k+1/.2}(C^n_{i,j,k} - C^n_{i,j,k+1}) \\ &= \left(C^n_{i,j,k} - C^{n-1}{}_{i,j,k}\right) \frac{\triangle x \triangle y \triangle z}{\triangle t} \end{aligned} \tag{6}$$

Next, if we define the constants

$$\begin{aligned} D_1 &= -\frac{\triangle y \triangle z}{\triangle x} D_{i-1/2,j,k} \quad D_2 = -\frac{\triangle x \triangle z}{\triangle y} D_{i,j-1/2,k} \quad D_3 = -\frac{\triangle x \triangle y}{\triangle z} D_{i,j,k-1/2} \\ D_4 &= -\frac{\triangle y \triangle z}{\triangle x} D_{i+1/2,j,k} \quad D_5 = -\frac{\triangle x \triangle z}{\triangle y} D_{i,j+1/2,k} \quad D_6 = -\frac{\triangle x \triangle y}{\triangle z} D_{i,j,k+1/2} \end{aligned} \tag{7}$$

and

$$E = \frac{\Delta x \Delta y \Delta z}{\Delta t} \tag{8}$$

equation 6 becomes

$$\begin{aligned} &D_1(C^n_{i,j,k} - C^n_{i-1,j,k}) + D_2(C^n_{i,j,k} - C^n_{i,j-1,k}) + D_3(C^n_{i,j,k} - C^n_{i,j,k-1}) - \\ &D_4(C^n_{i,j,k} - C^n_{i+1,j,k}) - D_5(C^n_{i,j,k} - C^n_{i,j+1,k}) - D_6(C^n_{i,j,k} - C^n_{i,j,k+1}) = \left(C^n_{i,j,k} - C^{n-1}{}_{i,j,k}\right) E \end{aligned} \tag{9}$$

Collecting like terms from equation 11 and multiplying both sides by -1 results in

$$\begin{aligned} &-D_1(C^n_{i-1,j,k}) - D_2(C^n_{i,j-1,k}) - D_3(C^n_{i,j,k-1}) + C^n_{i,j,k}(D_1 + D_2 + D_3 + D_4 + D_5 + D_6 + E) - \\ &-D_4(C^n_{i+1,j,k}) - D_5(C^n_{i,j+1,k}) - D_6(C^n_{i,j,k+1}) + C^n{}_{i,j,k} E = C^{n-1}{}_{i,j,k} E \end{aligned} \tag{10}$$

Equation 10 is an equation for node *i, j, k.* Deriving an equation for all $n = nxnynz$ nodes results in a set of *n* simultaneous equations being solved for *n* unknown concentration values. Equation 10 can be re-written as

$$\mathbf{A} \bullet \mathbf{c} = \mathbf{f} \tag{11}$$

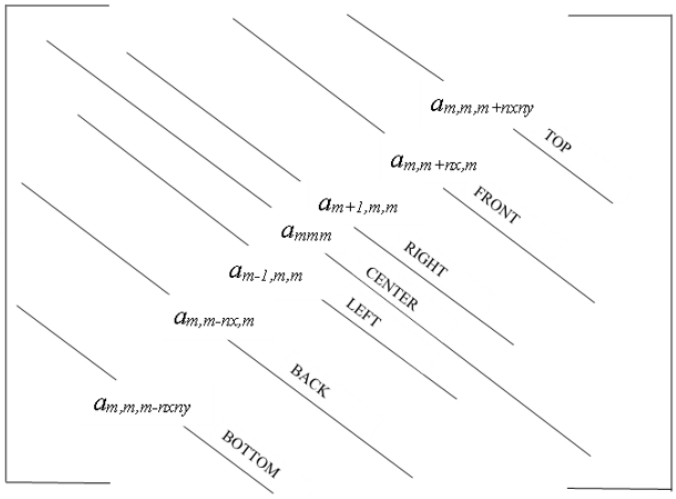
and its inverse as

$$\mathbf{c} = \mathbf{A}^{-1} \bullet \mathbf{f} \tag{12}$$

where $\mathbf{c}$ is the vector of unknown concentrations, $\mathbf{A}$ is the coefficient matrix that contains the values on the left hand side of equation 10, and $\mathbf{f}$ is the load vector that contains the values on the right hand side of equation 10. The coefficient matrix is a symmetric $mxm$, sparse banded matrix whose values depend on $\Delta x$, $\Delta y$, $\Delta z$, and *D.*

The matrix is a 7 point grid and defined as

A=




(13)

Where *m* is a global indexing scheme defined as

$$m = (k-1) \cdot nx \cdot ny + (j-1) \cdot ny + i \tag{14}$$

so that

$$C_{i,j,k} = C_m \tag{15}$$

The main diagonal (center) $a_{mmm}$ includes values from the left hand side of equation 12 and is defined as

$$a_{mmm} = (D_1 + D_2 + D_3 + D_4 + D_5 + D_6 + E) \tag{16}$$

The load vector $\mathbf{f}$ contains information about boundary conditions and source and sink terms. We automatically fill these locations with zeros. The modification to the coefficient matrix set up is later discussed. For boundary conditions, the load vector and coefficient matrix need to be modified by adding a defined constant to each side. For a constant concentration boundary on the top,

$$J_{in} = -2D\frac{\Delta x \Delta y}{\Delta z}(C^n_{i,j,k} - \text{bvT}) \tag{17}$$

where bvT is the top boundary value. The coefficient matrix is modified by

$$a_{mmm} = a_{mmm_{old}} + 2D\frac{\Delta x \Delta y}{\Delta z} \tag{18}$$

and the load vector by

$$f_m = f_{m_{old}} + 2D\frac{\Delta x \Delta y}{\Delta z} bvT \tag{19}$$

to reflect the boundary conditions. The new concentrations are then solved as before at time $t^n$.

Results from verification tests were compared to analytical solutions for simple cases of molecular diffusion. The performance of the program was evaluated by comparing the numerical model solution with analytical solutions from *Crank* (1975) for molecular diffusion and *Carslaw and Jaeger* (1959) for flow of heat in a rectangle. We solved the numerical model for a non-steady state diffusion in a one-dimensional (1D) plane sheet with uniform initial distributions and equal surface concentrations as well as heat flow in a two-dimensional (2D)

finite rectangle with faces having unit initial temperatures. The 1D and 2D models were subject to the following initial concentrations and boundary conditions

$$C(x,0) = 1.0$$
$$C(x,y,0) = 1.0$$
$$C = 0 \text{ for } x = \pm l$$
$$C = 0 \text{ for y} = \pm b$$

The one-dimensional solution in the form of a trigonometrical series is given by *Crank* (1975) equation 4.16 as

$$C = C_0 \frac{4}{\pi} \sum_{n=0}^{N} \left[ \frac{1}{2n+1} \sin\left[ \frac{2n+1\pi x}{l} \right] \exp\left[ -\frac{(2n+1)^2 \pi^2 Dt}{l^2} \right] \right] \quad (20)$$

where $C_0$ is the initial concentration, $l$ is distance from the middle, $x$ is the x coordinate location, $D$ is the effective diffusion coefficient, and $t$ is time. We set *x* equal to 58 which results in a *l* value of $\frac{x}{2}$. The normalized percent error between the numerical results and 1D analytical solution were less than 0.003 % (Figure A-3).

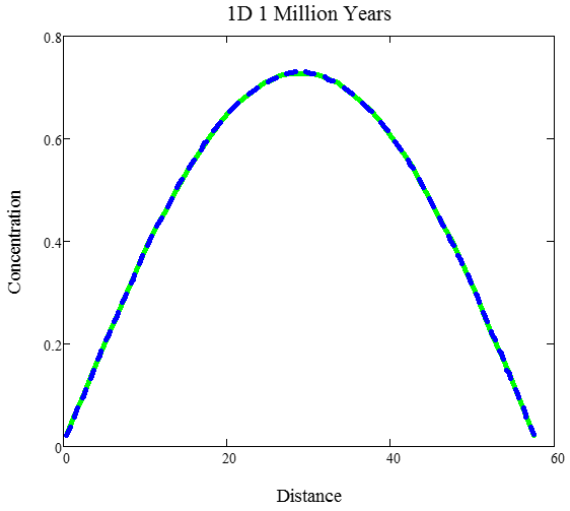



Figure A-3. 1D analytical verification. 1D analytical solution (green) from *Crank* (1975) equation 4.16 compared to numerical results (blue) of the diffusion code for an initial concentration of 1 over the entire domain.

The two-dimensional solution of *Carslaw and Jaegar* (1959) is given as

$$\psi(x,l) = \frac{4}{\pi}\sum_{n=0}^{N}\left[\frac{-1^n}{2n+1}\exp\left[\frac{-D(2n+1)^2\pi^2 t}{4l^2}\right]\cos\left[\frac{\left[(2n+1)\pi x\right]}{2l}\right]\right]$$
$$\psi(y,b) = \frac{4}{\pi}\sum_{n=0}^{N}\left[\frac{-1^n}{2n+1}\exp\left[\frac{-D(2n+1)^2\pi^2 t}{4l^2}\right]\cos\left[\frac{\left[(2n+1)\pi y\right]}{2l}\right]\right] \quad (21)$$

$$C = \psi(x,l)\psi(y,l) \quad (22)$$

where $\psi$ denotes the concentration as a function of *x*, *y*, *l*, and *b*. We set *l* equal to 98 and *b* equal to 28 to match our model dimensions. The normalized percent error between the numerical results and analytical 2D solution was less than 0.5 % (Figure A-4).

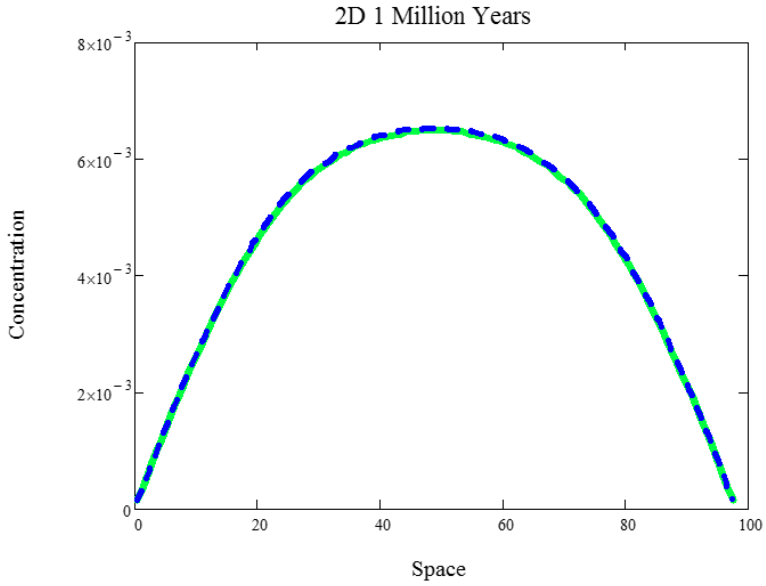



Figure A-4. 2D analytical verification. 2D analytical solution (green) from *Carslaw and Jaegar* (1959) p. 173 compared to numerical results (blue) of the diffusion code for an initial concentration of 1 over the entire domain.

Furthermore, the principle of conservation of mass requires that the net flux *i.e.* the cumulative sum of mass inflows and outflows must equal the accumulation of mass. The corresponding mass of a diffusing substance remaining after a given amount of time was compared to the analytical solution of *Crank* (1975) equation 4.18 which is

$$M = \sum_{n=0}^{N}\left[\frac{8}{(2n+1)^2\pi^2}\exp\left[\frac{-D(2n+1)^2\pi^2 t}{4l^2}\right]\right] \tag{23}$$

where, $M$ is the mass remaining in the system after a given time $t$. The percent error between the numerical results and the mass analytical solution was less than 0.1 % (Figure A-5).

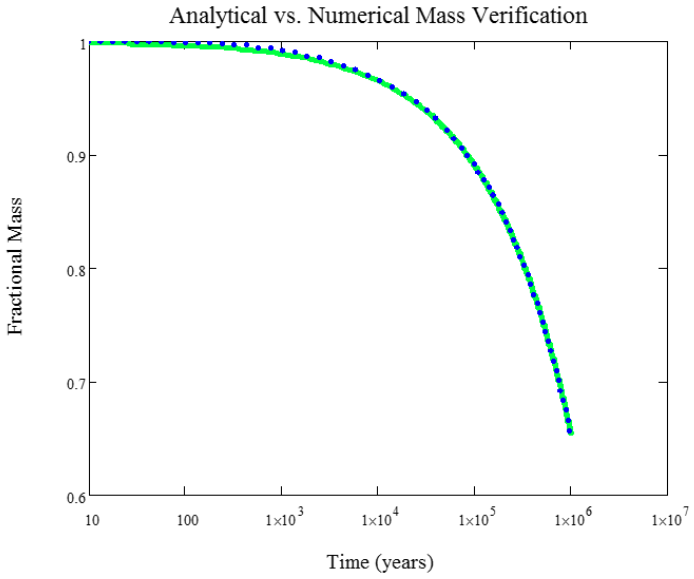



Figure A-5. Mass analytical verification. Numerical results from the diffusion code (blue) and analytical solution (green) from *Crank* (1975) equation 4.18. Approximately 65% of the air mass remains after 1 MY. Effective diffusion coefficient= $10^{-11}$m$^2$/s.

### *1.2 Hardware Platform*

The codes in this manual were developed on a DELL DESKTOP computer with the following specifications:

**Processor:** Intel® Core™ i7-3770 CPU @ 3.40 GHz

**RAM** = 8.00 GB

**Software System (OS):** Windows 7 Professional, 64 bit Operating System

**Compiler:** Intel® Visual Fortran Compiler Microsoft Visual Studio 10.0 2013

The code was developed to operate under a 64 bit WINDOWS environment, specifically for WINDOWS 7 OS. Installation of the software is done by simply copying the executable file **Diff_Perc.exe** (provided in the media file) into a designated file directory. All input and IMSL library files must be copied to the same directory. Double clicking the executable will run the program. The code can also be compiled and run on Microsoft Visual Studio 2013 in an x64 bit operating environment to make changes within the code or for debugging options. This method is preferred for accessing compilation errors that may occur. If a large domain (*e.g*. 100x30x60) is being used, the stack reserve size and stack commit size must be increased in DIFF_PERC under Visual Studio>Properties>Linker>System to the largest quantity possible to increase the total stack allocation size for the arrays in physical memory under the active x64 platform (Figure A-6). The diffusion code may be run on a Win32 operating environment if the size of the domain is fairly small (*e.g*. 10x10x10).

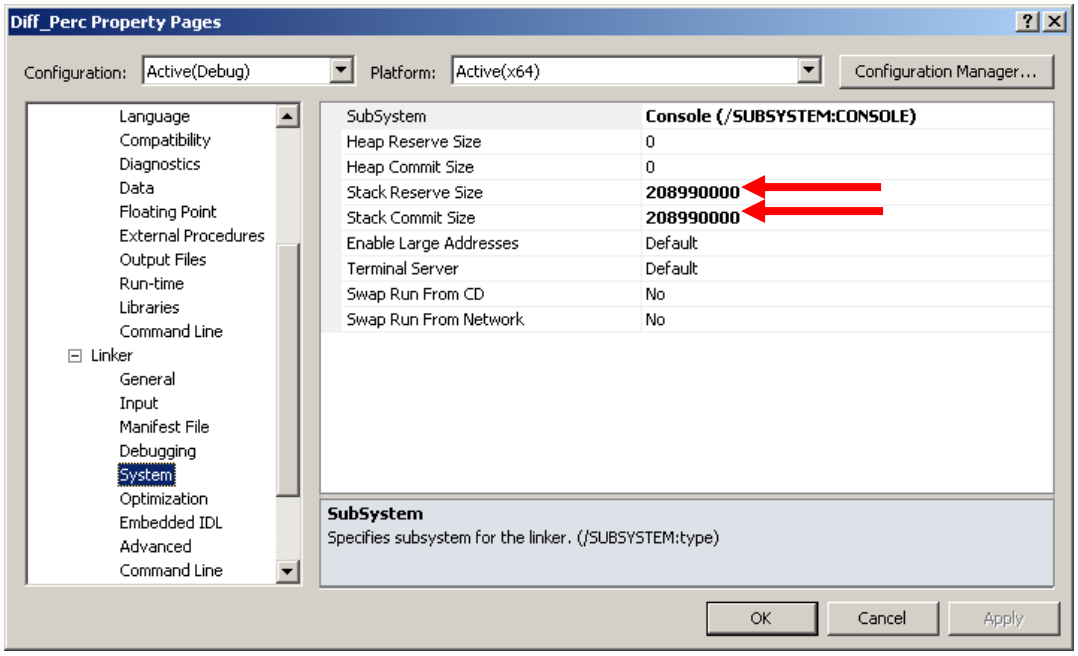



Figure A-6. DIFF_PERC property settings. Increasing the system linker for stack reserve and commit size for large domains.

## 2.0 DIFF_PERC Code Structure

The code structure, at a high level, describing the principal segments, is given here.

The main driver program is called **DIFF_MAIN.** It calls:

1. **DIFFIN** to read all input information and allocate memory.
2. **COEFF_MATRIX** to construct the coefficient matrix $\mathbf{A}$ in band storage.
3. **FVECTOR** to calculate the load vector $\mathbf{f}$ and modify the coefficient matrix $\mathbf{A}$ for specified boundary conditions.
4. **CLUSTBC** to locate and define active cells bounding cluster cells
5. **CSOLV** to solve $\mathbf{c} = \mathbf{A}^{-1} \bullet \mathbf{f}$ using a preconditioned conjugate gradient method.
6. **FLUXMASS** to calculate the net mass flux lost in the clusters. Updates cluster pressures.
7. **MBERROR** to calculate mass balance errors of total domain per time step.
8. **DIFFOUT** to write the final array of concentration values.

The **DIMEN** module declares and sets global variables used for all the above subroutines. It is used to reduce excessive variable declarations for each subroutine individually. The module contains specifications and definitions that can be used by one or more program subroutines. For the module to be accessible, the other program subroutines must reference its name in a **USE** statement. This code structure is shown in Figure A-7. A list of variable definitions can be found in Appendix A.

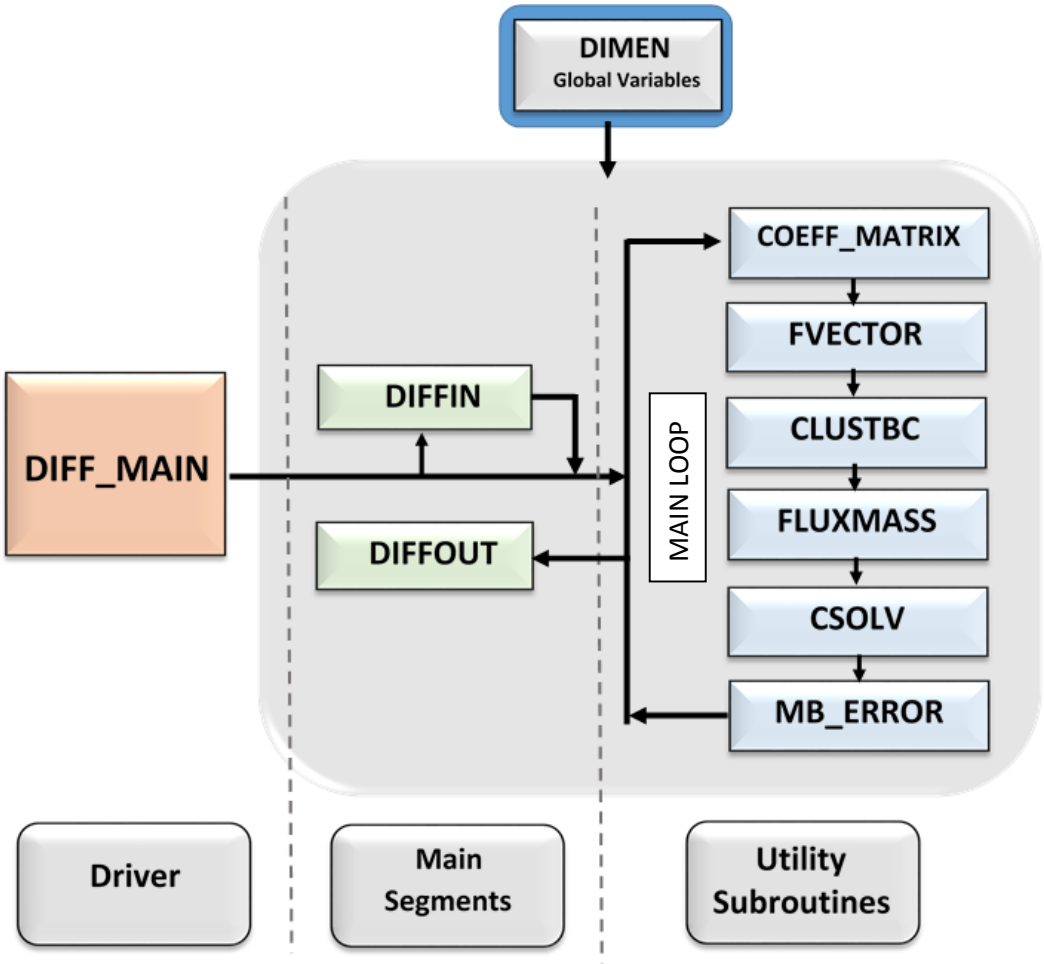



Figure A-7. DIFF_PERC code structure.

### *2.1 Input Driver: DIFFIN*

The diffusion code is incorporated into the MMIP3 model by using the trapped air cluster properties from the restart file. The main input subroutine **DIFFIN** is required to run the diffusion code and reads information from the main input file **diff1.inp** (Figure A-8) This file includes: a five character string (str); size of the simulation domain and depth of overburden pressure(water); size (widths) of the individual blocks (considered to be uniform across the model domain); the initial time step size; maximum time step size; porosity; number of total time steps, initial effective diffusion coefficient; free water diffusion coefficient; atmospheric pressure, molecular weight of gas used; Henrys dimensionless constant; time step scaling factor;

concentration output frequency; universal gas constant; maximum number of iterations; relative error value; logical variables for boundaries and concentration values for boundaries. The porosity is assumed to be constant and continuous across the domain. Also, the porosity term is canceled out in the final diffusion equation so is technically not needed. An array of initial pressures from the MMIP3 code are read in from a file called **rediff.out** which is the output from the **RESTART.f90 program** that will be later discussed. An example input file can be seen in Figure 8. **Itmax** and **relerr** are variables needed for the IMSL routine **PCGRC** in the **CSOLV** subroutine later discussed. These set the maximum number of iterations and relative error computed in the system solver. A user specified depth of water is appended to the initial concentration array. The initial air cluster pressures are adjusted to concentrations using equation 2 from *Holocher et al*., 2002

$$C_i = \frac{C_i^{gas}}{K_{H,i}(T,S)} = \frac{p_i}{R \cdot T \cdot K_{H,i}(T,S)} \tag{24}$$

where, $C_i$ is the equilibrium concentration of the gas $i$, $p_i$ is the partial pressure in the gas phase, $R$ is the universal gas constant, and $K_{H,i}$ is the dimensionless Henry coefficient which depends on the temperature $T$ and salinity $S$. A $K_{H,i}$ value of 63.5 was used for $N_2$ at 25° Celsius (*Holocher et al.,* 2002). Cells surrounding cluster cells are set to atmospheric concentration but may be defined according to equilibrium concentrations with depth. The initial mass in the domain and initial conditions for the mass balance error subroutine variables are defined for cumulative calculations. The coefficient matrix **A**(*i*, *j*) is zeroed out in the main program where *i* and *j* denote the row and column number, respectively, to reduce computational time and redundant set up for each time step. The initial concentrations and their corresponding cell indices are mapped to **InitialC.out** for visualization purposes. The number of cells that occupy each cluster are read in from **NumClust.out** and are included in the mass flux per volume calculation later. This file is also produced from **RESTART.f90**

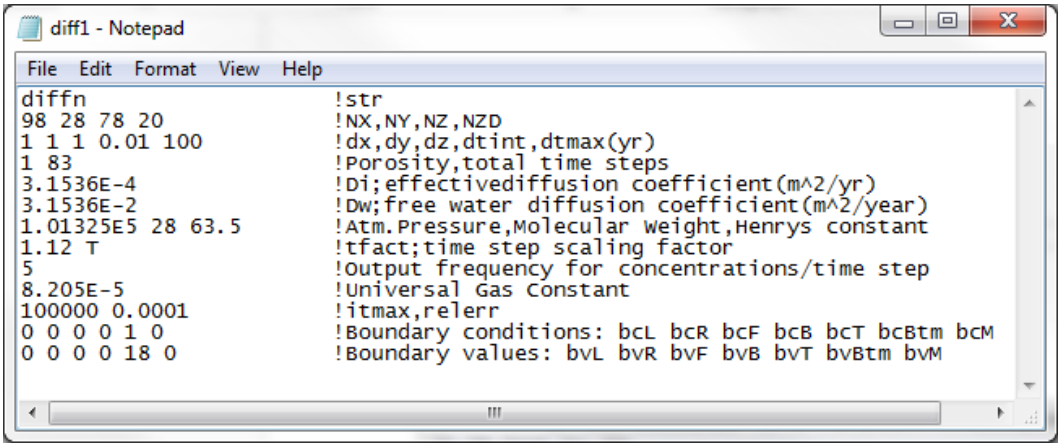



Figure A-8. Example input file. Input variables required.

### *2.1.1 Discretization of the Model Domain*

The model domain is discretized into a three dimensional cartesian grid, with *nx*, *ny* and *nz* number of grid blocks in the *x*, *y*, and *z* directions (Figure A-9). Each grid block is assumed to have the same dimension widths *dx*, *dy*, *dz* in the *x*, *y*, and *z* directions, respectively (Figure A-10). The indices. *i*, *j* and *k* are used to denote grid block numbers in the *x*, *y*, and *z* directions. The grid blocks (also called cells or nodes) at the two ends in each direction are considered inactive in the MMIP3 percolation code, so the dimensions used reflect the active sub-grid domain. Therefore, a 100 *x* 30 *x* 60 domain has sub-grid dimensions of 98 *x* 28 *x* 58. The grid block indices *m* are calculated over the entire grid including the active cells and inactive cells on all the boundaries. The global indexing scheme for each cell can be seen in (Figure A-11). *NZD* is the length (depth) of overburden pressure (water) that is appended to the top of the sub-grid domain in **DIFFIN**. Diffusion is possible from the (1) top, or (2) from sides that reflect 1$^{st}$ and 2$^{nd}$ type boundary conditions. An integer of 1 in **diff1.inp** implies a constant concentration boundary with specified values listed and a 0 represents a no flow boundary.

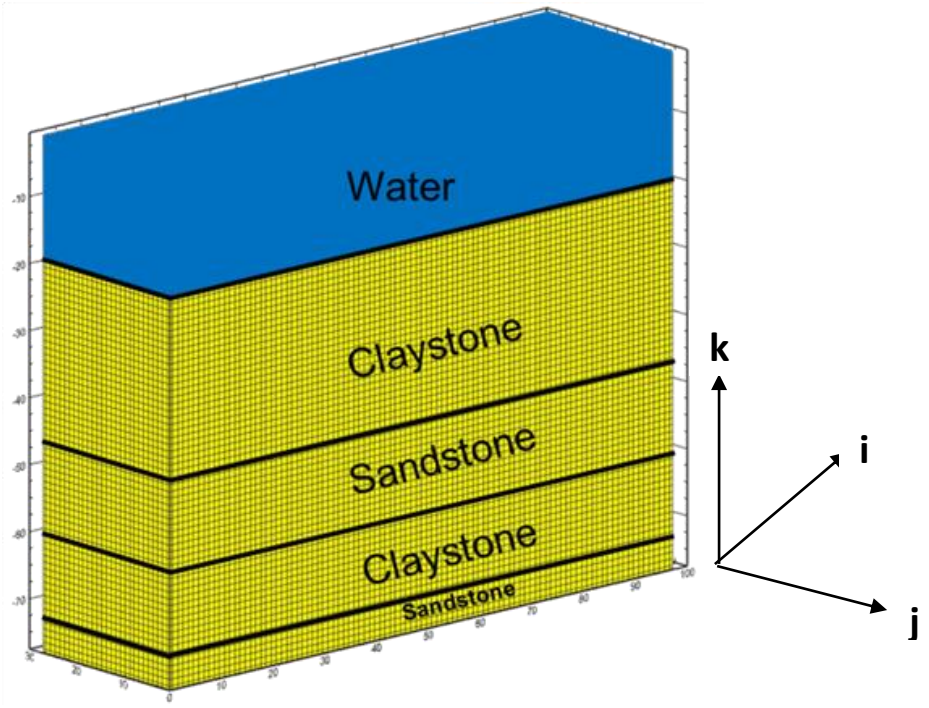



Figure A-9. Discretization of model domain.

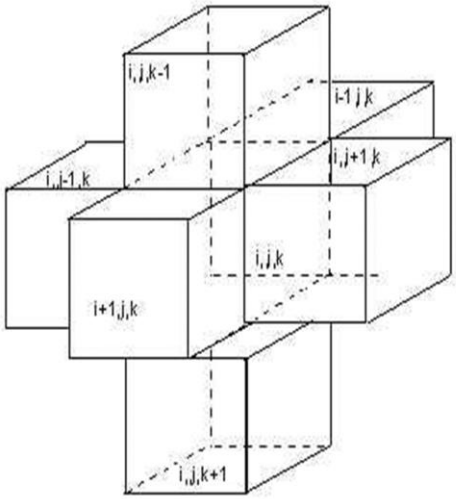



Figure A-10. 3D indexing scheme. (*McDonald and Harbaugh*, 1988.)

| | | | |
|---|---|---|---|
| 13 | 14 | 15 | 16 |
| 9 | 10 | 11 | 12 |
| 5 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 |

| | | | |
|---|---|---|---|
| 29 | 30 | 31 | 32 |
| 25 | 26 | 27 | 28 |
| 21 | 22 | 23 | 24 |
| 17 | 18 | 19 | 20 |

| | | | |
|---|---|---|---|
| 45 | 46 | 47 | 48 |
| 41 | 42 | 43 | 44 |
| 37 | 38 | 39 | 40 |
| 33 | 34 | 35 | 36 |

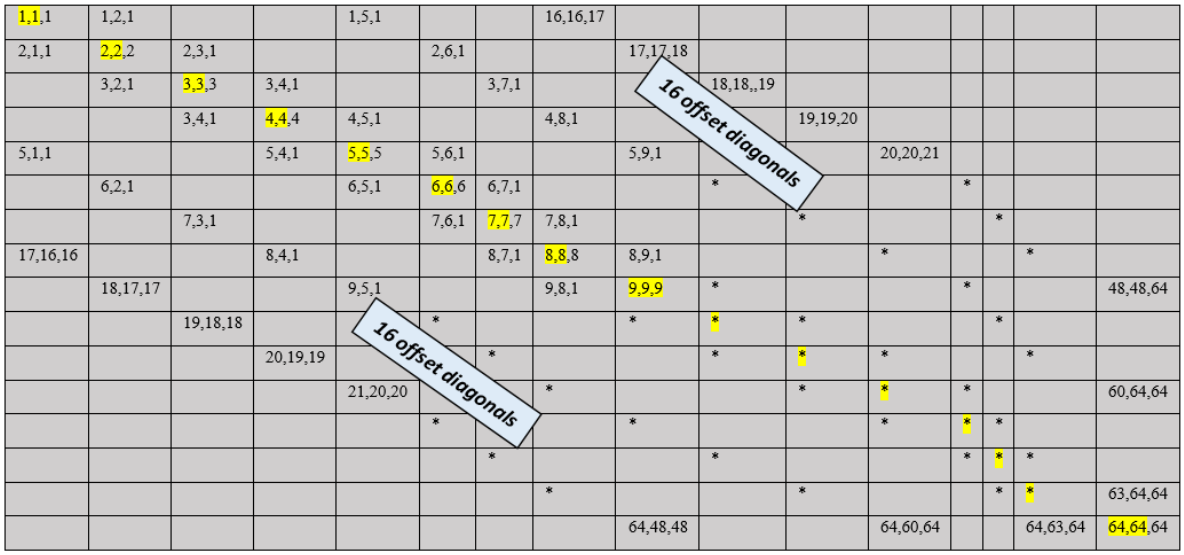
| | | | |
|---|---|---|---|
| 61 | 62 | 63 | 64 |
| 57 | 58 | 59 | 60 |
| 53 | 54 | 55 | 56 |
| 49 | 50 | 51 | 52 |

Figure A-11. Cell identification scheme. An example 4*x*4*x*4 domain global indexing scheme $m = (k-1) \cdot nx \cdot ny + (j-1) \cdot ny + i$

### *2.2 Coefficient Matrix Setup: COEFF_MATRIX*

The subroutine **COEFF_MATRIX** sets up the domain in a band storage structure. A band storage structure of the coefficient matrix saves both storage, space and computation time. A banded matrix is stored so the *j*th column of the matrix corresponds to the *j*th column of the array. This iterative technique requires less storage than direct methods for sparce matrices since the zero diagonals between the main diagonal and the outermost non-zero diagonals do not need to be stored. An illustration of the band storage construct can be seen in Figures A-12 and A-13. The storage is an *m*-by-*n* band matrix with three lower non-zero rows (left, front, top) and three upper non-zero rows (right, back, bottom). They represent the cell direction from the center nodes in the main diagonal and are stored in a compact two-dimensional array with *nxny+1* rows and *nnod* columns. Columns of the matrix are stored in the corresponding columns of the array, and diagonals of the matrix are stored in the corresponding rows of the array.

First, the conditions for all sides of the domain must be defined. For example, for all nodes where $i$ =1, the condition on the left (**cndL**) is defined as zero since there are no grid blocks beyond the boundary. For all other nodes, the boundary conditions are defined as the average effective diffusion coefficient or harmonic mean. Next, the coefficient matrix is constructed in band storage. The rows and columns in matrix **A** are built to reflect the conditions for all sides of the domain, including the main diagonal. The structure is built for the water and subgrid since they have different diffusion coefficients. The offset between the rows in band storage reflect the same number of zero diagonals that would occur in a regular sparce *mxm* matrix. This technique essentially rotates the diagonal matrix to reduce the number of zero elements in a smaller storage spectrum.




| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1,1,1 | 1,2,1 | | | 1,5,1 | | | 16,16,17 | | | | | | | | |
| 2,1,1 | 2,2,2 | 2,3,1 | | | 2,6,1 | | | 17,17,18 | | | | | | | |
| | 3,2,1 | 3,3,3 | 3,4,1 | | | 3,7,1 | | | 18,18,,19 | | | | | | |
| | | 3,4,1 | 4,4,4 | 4,5,1 | | | 4,8,1 | | | 19,19,20 | | | | | |
| 5,1,1 | | | 5,4,1 | 5,5,5 | 5,6,1 | | | 5,9,1 | | | 20,20,21 | | | | |
| | 6,2,1 | | | 6,5,1 | 6,6,6 | 6,7,1 | | | * | | | * | | | |
| | | 7,3,1 | | | 7,6,1 | 7,7,7 | 7,8,1 | | | * | | | * | | |
| 17,16,16 | | | 8,4,1 | | | 8,7,1 | 8,8,8 | 8,9,1 | | | * | | | * | |
| | 18,17,17 | | | 9,5,1 | | | 9,8,1 | 9,9,9 | * | | | * | | | 48,48,64 |
| | | 19,18,18 | | | * | | | * | * | * | | | * | | |
| | | | 20,19,19 | | | * | | | * | * | * | | | * | |
| | | | | 21,20,20 | | | * | | | * | * | * | | | 60,64,64 |
| | | | | | * | | | * | | | * | * | * | | |
| | | | | | | * | | | * | | | * | * | * | |
| | | | | | | | * | | | * | | | * | * | 63,64,64 |
| | | | | | | | | 64,48,48 | | | 64,60,64 | | | 64,63,64 | 64,64,64 |

Figure A-12. Sparce matrix. A sparse 4x4x4 3D matrix structure with 4 diagonals between the main diagonal and back/front diagonals and 16 empty diagonals (not shown) between front/top and back/bottom. Blanks = 0.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 16,16,17 | 17,17,18 | 18,18,19 | 19,19,20 | 20,20,21 | * | * | * | * | * | * | 48,48,64 |
| | | | | | | | | | | | | | | | |
| | | | | | | *11 rows not shown here | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | 1,5,1 | 2,6,1 | 3,7,1 | 4,8,1 | 5,9,1 | * | * | * | * | * | * | 60,64,64 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | 2,1,1 | 3,2,1 | 4,3,1 | 5,4,1 | 6,5,1 | 7,6,1 | 8,7,1 | 9,8,1 | 10,9,1 | * | * | * | * | * | 64,63,64 |
| 1,1,1 | 2,2,2 | 3,3,3 | 4,4,4 | 5,5,5 | 6,6,6 | 7,7,7 | 8,8,8 | 9,9,9 | * | * | * | * | * | * | 64,64,64 |
| 1,2,1 | 2,3,1 | 3,4,1 | 4,5,1 | 5,6,1 | 6,7,1 | 7,8,1 | 8,9,1 | * | * | * | * | * | * | 63,64,64 | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| 5,1,1 | 6,2,1 | 7,3,1 | 8,4,1 | 9,5,1 | * | * | * | * | * | * | 64,60,64 | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | *11 rows not shown here | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| 64,48,48 | * | * | * | * | * | * | 20,20,21 | 20,19,19 | 19,18,18 | 18,17,17 | 17,16,16 | | | | |

Figure A-13. Band storage. A 4x4x4 3D coefficient matrix in band storage forms *2nxny+1* rows and *nxnynz* columns. The yellow highlighted row is the main diagonal of the matrix. * are continued values to the end. Blanks are zero.

### *2.3 Load Vector Modification: FVECTOR*

The subroutine **FVECTOR** sets up the load vector $\mathbf{f}$ of the system of equations and modifies the coefficient matrix $\mathbf{A}$ and load vector for the specified boundary conditions as seen in equations 17-19. The main diagonal of the matrix is first calculated followed by the other domain sides. The main diagonal represents the right hand side of equation 3 and is calculated for every node. Integer variables from the **diff1.inp** file designate if a boundary needs to be accounted for to reflect the type and value of that boundary. For example, if **bcL** is set to 1 in the input file then the concentrations on the left side of the domain are modified to the boundary value input in equation 18. An integer variable of 0 means the nodes for that boundary are not modified and are considered a no flow boundary. The preconditioned part of the matrix **PC** is modified the same way.

### *2.4 Defining Cluster Boundaries: CLUSTBC*

A cluster is a group of cells that have an interconnected air phase and are deemed inactive in the diffusion model. This subroutine locates active cells bounding cluster cells and identifies which boundary sides of those cells are adjacent to the cluster cell. A cluster cell completely bounded by other cluster cells (*i.e.* the center of a cluster) are not indexed. An indexing scheme is first defined for all the active cells adjacent to a cluster cell for the left **CBL**, right **CBR**, front **CBF**, back **CBB**, top **CBT**, and bottom **CBBTM** of each cell. If a cell in the cluster index array **Cindx (Ccell)** defined in **diff1.inp** has a value of 1 then it is a cluster cell and 0 if it is an inactive cluster cell. The first portion of the subroutine traverses across the domain and locates the cells with a value of 1 (cluster cell) and defines the indexed boundary conditions for that cell. For example, if a cluster cell has active cells to the left and right then the boundary condition cells to the left **CBL** and right **CBR** of that cluster cell are set to 1. These values will be used in calculating the net mass flux out of an entire cluster in the **FLUXMASS** subroutine. All cluster cells on the boundary nodes of the domain are accounted for next. If a cluster cell is located on the left edge boundary of the domain then it cannot have a boundary condition cell to the left of it and **CBL** = 0. This rule is applied to the other domain boundaries. If a cluster cell is bounded by another cluster cell on any side then an indexed boundary condition is not defined for that side of the cell. A cluster cell surrounded by cluster cells on all 6 sides will have a value of 0 defined for the

boundary cell index condition for all sides. Finally, the load vector $\mathbf{f}$ and coefficient matrix $\mathbf{A}$ are modified for those active cells that have boundary conditioned values of 1 for each side and equations 17, 18, and 19 become

$$J_{in} = -2D\frac{\Delta x \Delta y}{\Delta z}(C^{n}_{i,j,k} - C_P) \tag{25}$$

$$a_{mmm} = a_{mmm_{old}} + 2D\frac{\Delta x \Delta y}{\Delta z} \tag{26}$$

$$f_m = f_{m_{old}} + 2D\frac{\Delta x \Delta y}{\Delta z}C_P \tag{27}$$

where $C_p$ is the pressure in the cluster cell adjacent to the active wetted cell. The subroutine ends and indexed boundary conditions for the cluster cells are past to the next subroutine (Figure A-14.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | * **3** | 0 |
| 1 | * | * | 1 | * **2** | * | 1 | 1 | * | 0 |
| * **1** | * | * | * | 0 | 0 | 1 | * | 0 | 0 |
| 0 | 0 | 1 | * | 0 | 0 | 1 | * **4** | 0 | 0 |

Figure A-14. Cluster index. Example indexing scheme for defining cluster boundary conditions to the left of a cluster cell**.** There are 4 clusters here (red). A 1 is a defined for a cluster cell that has an adjacent cell to the left. The other sides are accounted for in the same indexing loop.

### *2.5 Linear System of Equations: CSOLV*

This subroutine solves the system $\mathbf{A} \bullet \mathbf{c} = \mathbf{f}$ using the preconditioned conjugate gradient method (*e.g. Golub and Van Loan*, 1983), where $\mathbf{A}$ is the band storage coefficient matrix, $\mathbf{f}$ is the load vector, and *c* is the unknown concentration vector. IMSL subroutines are required (*IMSL FNL 7.0.1*). The preconditioning technique transforms the matrix equation, and then the generalized conjugate gradient method is applied to the transformed matrix. A USE_IMSL_LIBRARIES command must be inserted at the beginning of the routine. There are two ways to specify the set of libraries to link with. One is to use a predefined environment variable on the command line that links the application. The other is to add an INCLUDE line in one of the Fortran sources. The INCLUDE line method is more convenient and works in both the command line and Visual Studio environments, therefore, the line INCLUDE 'link_fnl_shared.h' is used. Installing and using IMSL subroutines depends on the type of compiler and software version being used and is beyond the scope of this manual.

The IMSL subroutine LFCQS factors the matrix and checks for non-positive definiteness or ill-conditions. The IMSL subroutine PCGRC solves a real symmetric definite linear system using a preconditioned conjugate gradient method (*Hageman and Young*, 1981) by utilizing two other IMSL operations. The first is MURBV which multiplies the real banded matrix in band storage mode $\mathbf{A}$ by the real vector $\mathbf{f}$. Then, LSLQS solves the real symmetric definite system of linear equations $\mathbf{c} = \mathbf{A}^{-1} \bullet \mathbf{f}$ in band storage mode without iterative refinement. Finally, the newly calculated concentrations are set to **Cold** and cycled through the rest of the program in the next time step. The last portion of this subroutine outputs the concentrations per a user defined time step **Tstep** for post processing visualization purposes.

### *2.6 Flux out of Clusters: FLUXMASS*

Concentrations across a cluster are distributed evenly. All cluster cells have the same defined concentration as the cluster itself. For example, if a cluster occupies 16 cells and has a concentration of 145 mg/L, then every cell in that cluster has a concentration of 145 mg/L. The same boundary indexing scheme used in **CLUSTBC** is applied to locate cluster cells bounded by active (wetted) cells for all directions and to calculate the mass flux out of each cluster cell on all 6 sides. If a boundary condition cell from before *i.e.* **CBT (indxT)** has a value of 1 then the mass flux out of the top of that cluster cell is calculated. The flux is the change in concentration across the cluster cell and its adjacent bounded active cell for all sides (Figure A-15) and is given as

$$FL_n = -2\frac{D}{\Delta x}\left(C_n(indxL) - C_P\right)\Delta y \Delta z \Delta t \tag{28}$$

where $FL_n$ is the flux out of the left side of the cluster cell *n*, $C_m(indxL)$ is the concentration of the active cell to the left of the cluster cell, and $C_p$ is the cluster cell concentration. The total flux out of each cluster is calculated by

$$Fsum_n = FL_n + FR_n + FF_n + FB_n + FT_n + FBTM_n \tag{29}$$

where $Fsum_n$ is the total flux out of cluster *n*, and $FL_n$, $FR_n$, $FF_n$, $FB_n$, $FT_n$, $FBTM_n$ are the fluxes out of the left, right, front, back, top, and bottom of the cluster, respectively. The mass in each cluster is calculated by

$$MassC = C_P \cdot \Delta x \Delta y \Delta z \cdot Ncell_n \tag{30}$$

where $MassC$ is the total mass in the cluster and $Ncell_n$ is the number of cells in the cluster. The sum of these mass fluxes are then subtracted from their original (previous) mass of the cluster. The total flux out of each cluster is given by

$$M_{new} = MassC - Fsum_n \tag{31}$$

where $M_{new}$ is the new mass reflecting the mass lost. Finally the new cluster concentration is calculated as

$$C_{Pnew} = \frac{M_{new}}{\Delta x \Delta y \Delta z} \cdot Ncell_n \tag{32}$$

These cluster concentrations reflect the change in mass per cluster and are defined as the new cluster cell concentrations for the next time step. The code calculation can be seen in Figure A-16.

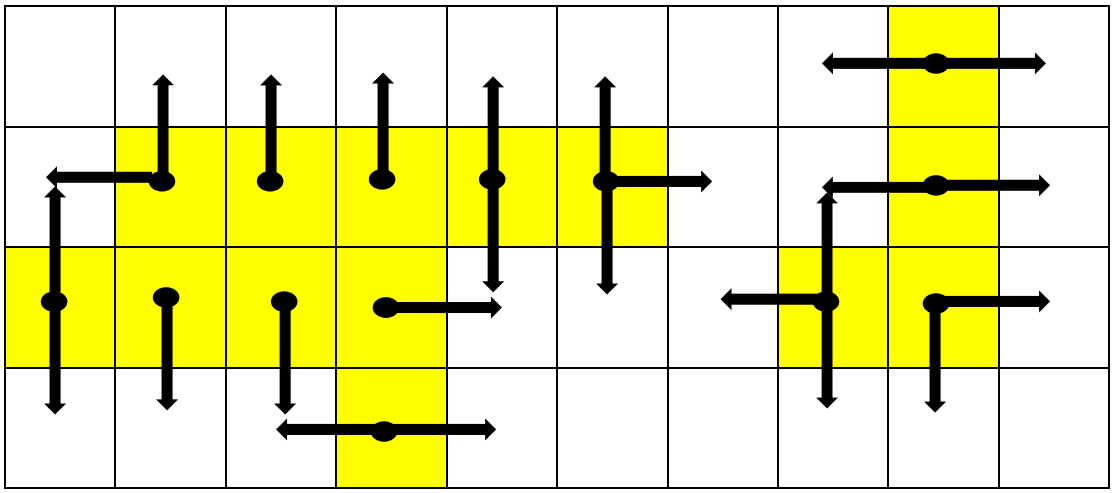


Figure A-15. Cluster mass flux. The total mass flux out of these two clusters (yellow) is the sum of the fluxes (arrows) out of every cluster cell adjacent to an active cell (white). This figure does not include the adjacent upper and lower layers.

```
!Sum up total fluxes out of the cluster:Some fluxes may be zero i.e the cluster center cell.
 Fsum(n)=FL(n)+FR(n)+FF(n)+FB(n)+FT(n)+FBTM(n)
 MassC(n)=ClustC(n)*(phi*dV)*Ncell(n)
 MassNew(n)=abs(MassC(n)-(Fsum(n)))
 Cnew(n)=MassNew(n)/(phi*dV*Ncell(n))
 !Define new clust concentrations
 ClustC(n)=Cnew(n)
```

Figure A-16. Code calculation of mass flux. Mass flux calculation for the cluster cells bounded by active cells.

### *2.7 Mass Balance Errors: MB_ERROR*

**MB_ERROR** outputs mass balance errors and conditions each time step. The principle of the conservation of mass requires that the net mass flux per time step must equal the net change in mass within the domain during that time step. The change in flux in with the change in flux out should have a small relative error. First, the flux out of each domain boundary is calculated. For example, the flux out of the top is calculated by

$$JT_m = \frac{-2D}{\Delta z}\left(C_m - bvT\right)\Delta x \Delta y \Delta t \tag{33}$$

$$JTT = JTT + JT_m \tag{34}$$

where, $JT_m$ is the flux across the top boundary, $C_m$ is the concentration for the top nodes, $bvT$ is the boundary value at the top, and $JTT$ is the total sum of fluxes for each node on the top boundary. The other sides of the domain are calculated in the same manner. The fluxes across each boundary in the domain are summed by

$$M_f = JLT + JRT + JFT + JBT + JTT + JBMT \tag{35}$$

where $M_f$ is the net flux, $JLT$ is the total flux out of the top, $JRT$ is the total flux out of the right, $JFT$ is the total flux out of the front, $JBT$ is the total flux out of the back, $JTT$ is the total flux out of the top, and $JBMT$ is the total flux out of the bottom. Next, the total mass in the domain is calculated by

$$M_T = M_T + M_D \tag{36}$$

$$M_D = C_m \Delta x \Delta y \Delta z \tag{37}$$

where $M_D$ is the mass of each node *m*, and $M_T$ is the total mass in the domain. $M_T$ =0 for the first loop. The change in mass of the domain is calculated by

$$M_{cs} = M_T - M_{Told} \tag{38}$$

where $M_{cs}$ is the net change in mass in the system, and $M_{Told}$ is the mass in the domain from the previous time step. The mass balance error is calculated by

$$M_{BE} = M_f - M_{cs} \tag{39}$$

$M_{BE}$ will be negative when there is more gas exiting across the boundaries than is lost from the total domain each time step. The normalized mass balance error is

$$M_{NE} = \frac{M_{BE}}{M_{cs}} \tag{40}$$

A mass residual error is computed as

$$M_{err} = \frac{M_{BE}}{M_0 - M_{cs}} \tag{41}$$

where $M_o$ is the initial mass in the domain. Cumulative mass balance errors throughout the compilation are kept track of as well. They include a cumulative mass balance error and a normalized cumulative mass balance error given as

$$M_{BEC} = M_{BE} + M_{BE}old \tag{42}$$

$$M_{NBEC} = M_{NE} + M_{NE}old \tag{43}$$

where $M_{BE}old$ is the mass balance error from the previous time step and $M_{NE}old$ is the normalized mass balance error from the previous time step. The cumulative mass out of the domain is calculated by

$$M_{cumul} = M_{cs} + M_{cs}old \tag{44}$$

where $M_{cs}old$ is the change in mass from the previous time step. A mass cumulative error is calculated by

$$M_{cerr} = \frac{M_{BEC}}{M_0 - M_{cumul}} \tag{45}$$

The normalized values of $M_{NE}$ at the end of test simulations were approximately ±1 %. Other mass calculations for each time step include a fractional mass leaving the domain

$$M_{resid} = \frac{M_{cumul}}{M_0} \tag{46}$$

And the fractional mass (residual) left in the domain as

$$1 - M_{resid} \tag{47}$$

These parameters are written to **MBerror.out** and **Mresidual.out** for every time step. The last portion of this subroutine updates the newly calculated mass variables and defines the size of the next time step **dt.** A Mathcad file **TotalTime.xmcd** has been supplied in the program directory to assist the user in calculating the desired total time (e.g days, years) based on variable time step and maximum time step sizes. The cumulative time past is calculated by

$$time = dt + dt_{old} \tag{48}$$

where $dt_{old}$ is the previous time step size. For a lengthy scenario (e.g 1 MY), the initial time step is multiplied by a time scale factor until the maximum time step is reached by

$$dt_{old} = dt \cdot tfact \tag{49}$$

$$dt = dt_{old} \tag{50}$$

where $tfact$ is the time scale factor defined in the input. Once the maximum desired time step is reached, each time step for the rest of the program is defined as $dt_{\max}$ by

$$\begin{aligned} &if\,(dt > dt_{\max})then \\ &dt = dt_{\max} \end{aligned} \tag{51}$$

where $dt$ is the time step for the rest of the program. This reduces computational run time.

### *2.8 Concentration Output file: DIFFOUT*

After the main loop of the program is complete, the concentrations are converted back in to their original pressure dimensions and the final array of pressures is outputted to **diff.out.** This routine also has the option to output the grid center location of each node in the *x*, *y*, and *z* directions. The cluster cells are inserted back in to their original indexed cell location defined at the beginning of the program and the other active cells are indexed to their corresponding cell number in the global indexing scheme. The file **diff.out** is the main input file for reconstructing the initial restart file used in the percolation code (mentioned later). The total domain that includes the depth of water is outputted to **difftotal.out** and is useful for visualization practices.

### *2.9 Program Driver: DIFF_MAIN*

The main driver of the program calls the subroutines previously discussed, redefines initial conditions, and allocates all of the arrays. It is simply an organized sequence of call statements to the primary subroutines (Figure A-17). After the main loop of the program and final output of concentrations, all of the arrays are deallocated to release the allocated memory and the program is ended. Mathcad files have been included in the program directory to compare the numerical results and mass remaining in the domain to 1D and 2D analytical solutions.

```
!*****************************************************************************************************
!Iterative counting loop for entire domain and # of time steps.
do kk=1,Nts

   !Add boundary conditions(Load Vector and Coeffcient Matrix)
   call coeff_matrix(A1,PC1,dt,ClustN)

   !Build the load vector f for entire domain
   call fvector(A1,PC1,f1,Cold,dt,ClustN)

   !Index cluster cells and modify load vector f and matrix A
   call clustBC(f1,PC1,A1,dt,ClustN,ClustP,Ccell,CBL,CBR,CBF,CBB,CBT,CBBTM)

   !Solve resulting system of equations for unknown concentrations
   call csolv(A1,PC1,f1,Cold,Cu,kk,TS,ClustN,ClustP)

   !Calculate total flux out of each cluster. Define new cluster pressures
   call fluxmass(Cold,ClustN,Ncell,ClustP,CBL,CBR,CBF,CBB,CBT,CBBTM,dt)

   !Calculate mass balance error parameters for each time step
   call mb_error(kk,Cold,Mcumul,time,dt,NMBE,Merror,Mresid)

    !Echo time steps to the screen
    write(*,*)kk,dtnew

end do
```

Figure A-17. Driver loop. The main loop that calls each subroutine and drives the program.

## 3.0 SUBPROGRAMS

There were two programs developed to couple the diffusion code with the MMIP3 code. These programs use values from the MMIP3 restart file and are compiled before and after the diffusion code. A random field generator program uses site specific parameters to generate realizations of air entry pressures. The structure for all the programs including DIFF_PERC can be seen in Figure A-18.

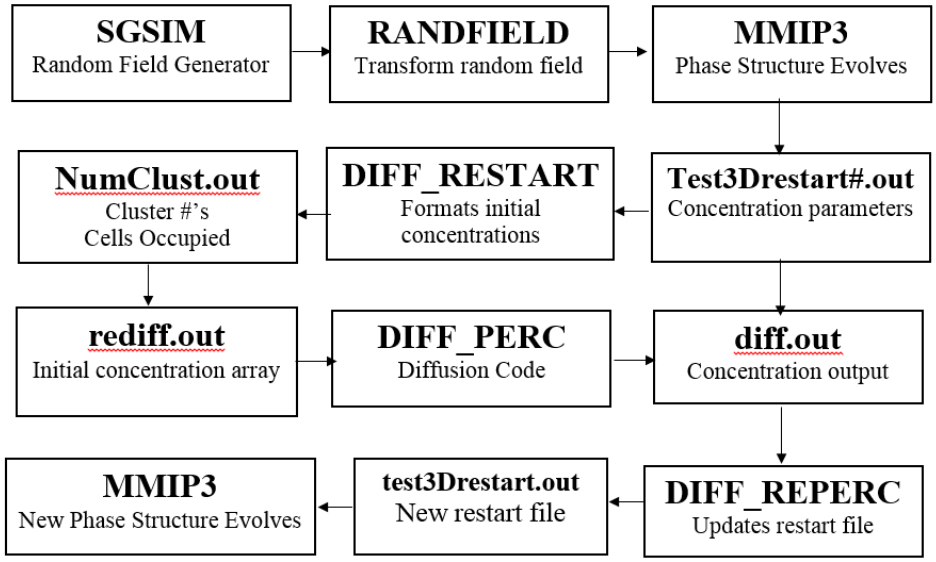



Figure A-18. Structure of coupling codes. Sequence of programs to superimpose the diffusion code with the percolation code.

### *3.1 Sequential Gaussian Simulation: SGSIM*

Initial water entry (spanning) pressures are determined by randomly sampling the distribution of observed air-entry pressures from a low-permeability mudrock. Both correlated and uncorrelated random fields of spanning pressures were generated. The random fields used for the initial MMIP3 compilation may be generated over a grid equal to or bigger than the active grid used for the percolation simulation. All the inputs are list directed i.e. free format. Correlated random fields (one random variable for one grid block) are generated over all cells in the active model domain to simulate the water entry pressures (spanning pressures). The random field files **Test3exp1_*.rv** are generated using the program **SGSIM** from GSLIB90 software (*Deutsch et al*., 1998) where * represents the layer number in the domain. These values generated by SGSIM

represent a random field (isotropic/anisotropic) with a prescribed correlation structure. Detailed parameters and explanation of SGSIM can be found at the end of this manual.

### *3.2 Random Field set up: RANDFIELD*

This program transforms the random fields from SGSIM to the correct input format and values needed for the MMIP3 code. The random values from the SGSIM program are outputted as mean zero. The random fields reflect a 4 layer correlated anisotropic domain with an exponential covariance structure. Air-entry (spanning) pressures are assumed to follow a log-normal distribution with geometric mean values of 2.2 and 1.8 MPa (log means of 8.65 and 5.95) and log variances of 2.7 and 1.1 for clay and sand, respectively (Figure A-19). These statistical values were obtained from site specific air-entry pressure experiments and subsurface discontinuity mapping reports (*Holt et al.*, 2011). RANDFIELD reads in the SGSIM output array**,** adds the means specified for clay or sand, and then log transforms them. The executable and code structure must be user changed for each random field structure. Inputs in **mean.inp** include the mean values for clay and sand and the domain size for that fraction of the total domain. The *nz* value reflects the length (depth) of the correlated structure to be used in the MMIP3 code. The code must be run for each sub layer of the domain. The user must specify in the code whether or not the mean values of clay and sand are being used (meanc and means), the thickness (length) *nz* in **mean.inp** and the corresponding output file names from SGSIM. For a simplistic approach, each random field input file from SGSIM has been renamed **sgsim1.out** for the first domain layer, **sgsim2.out** for the second layer and so on. The output is a file named **Test3exp1.rv** which can then be used in the percolation code. A sub _# declaration must be included to specify which layer or random field is being outputted. For example, the second layer in the random field domain needs to be named **Test3exp1_2**. The naming convention and character titles may be changed by the user in the percolation code if desired. These files must be in the same directory the percolation code is run. The quantity and lengths of the additional layers must be defined in the MMIP3 input file **Test3D_full.mip**.

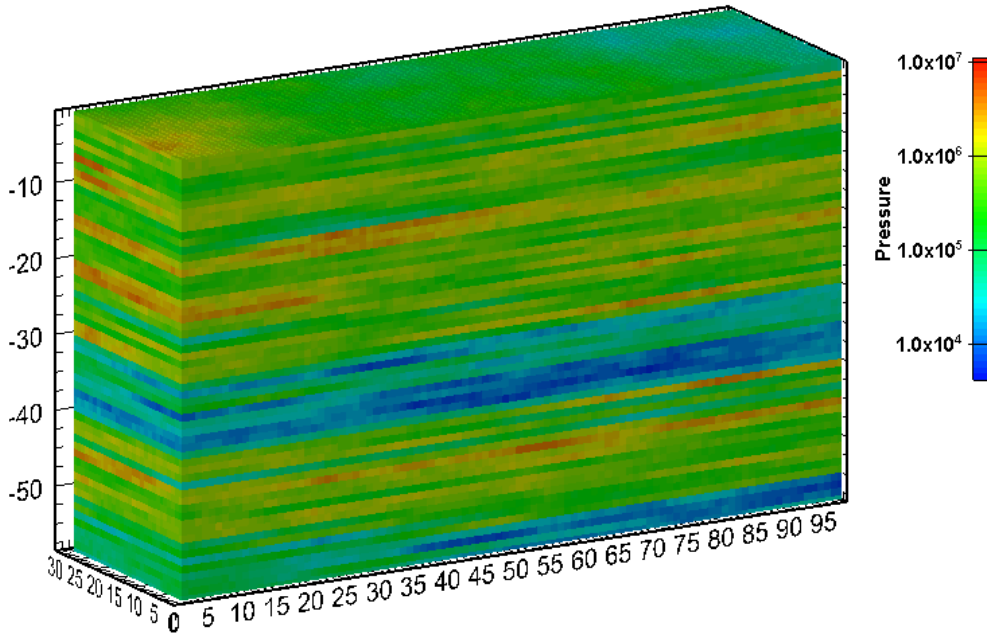



Figure A-19. Correlated random field. Example random field input for variable spanning pressures.

### *3.3 Initial Concentrations: DIFF_RESTART*

This program reads the domain dimensions from **dimen.inp** and pressures from the percolation codes final restart file *i.e.* **test3Drestart150788.out**. The number in the restart file name represents the final invasion step number in the percolation sequence. The user must specify this restart file name in the open statement at the beginning of the program (Figure A-20). The program first indexes the cluster pressures to their cell id number. Next, the partial invasion factors need to be modified for fully occupied cells from a value of 0 to 1. The pressures for each cluster cell are multiplied by their corresponding partial invasion factor because the diffusion code assumes the 1 $m^3$ grid blocks are fully and not partially occupied by air. Multiplying by the partial invasion factors decreases the cell pressures so that each 1 $m^3$ cluster cell is fully occupied by gas. Then, the cluster cell pressures are averaged across the number of cells the cluster occupies. This pressure array represents the sub-grid domain beneath the overburden water pressure where zeros represent active cells. These pressures are written to

**rediff.out** and are used as the initial concentration array for the DIFF_PERC program. An array that lists each cluster number and the number of cells it occupies is outputted to **NumClust.out.** This file is used in the diffusion code for averaging the mass flux out of a cluster.

### *3.4 New Concentrations: DIFF_REPERC*

**DIFF_REPERC** reads in the pressures from the diffusion code file **diff.out** and imports them back in to the restart file used for the MMIP3 restart simulation of gases expanding and removal of overburden pressure. Remember, the domain in the diffusion code is the sub-grid domain because the outer boundary cells of the MMIP3 code are inactive. Therefore, an inactive top layer of nodes needs to be appended to the **diff.out** file to match the original domain size of the initial restart file. The user must specify which restart file is being updated in the open statement at the beginning of the program (Figure A-20). The pressures are read using the same global indexing scheme that was used in the percolation code, cycling on *k*, *j*, and then *i*. Next, the program indexes the cluster pressures from the diffusion code with their coinciding **cell ID** location and averages them based on their partial invasion factor and number of cells (cell volume) **NumCell** the cluster occupies. For cluster cells that are partially invaded, the pressures are divided by their original partial invasion factor to 'squeeze' the pressures in to their original partial volume occupied. For clusters larger than one cell, the pressures are averaged across the cluster volume occupied. The output file **test3Drestart.out** is a mirror image of the initial restart file except for the newly calculated and reduced cluster pressures. It is used as the initial restart input file for the MMIP3 code and must be user defined in the MMIP3 input file **Test3D_full.mip** (Figure A-21). Differences in saturation and other hydrologic parameters can then be analyzed and compared that reflect the effects of diffusion on the system.

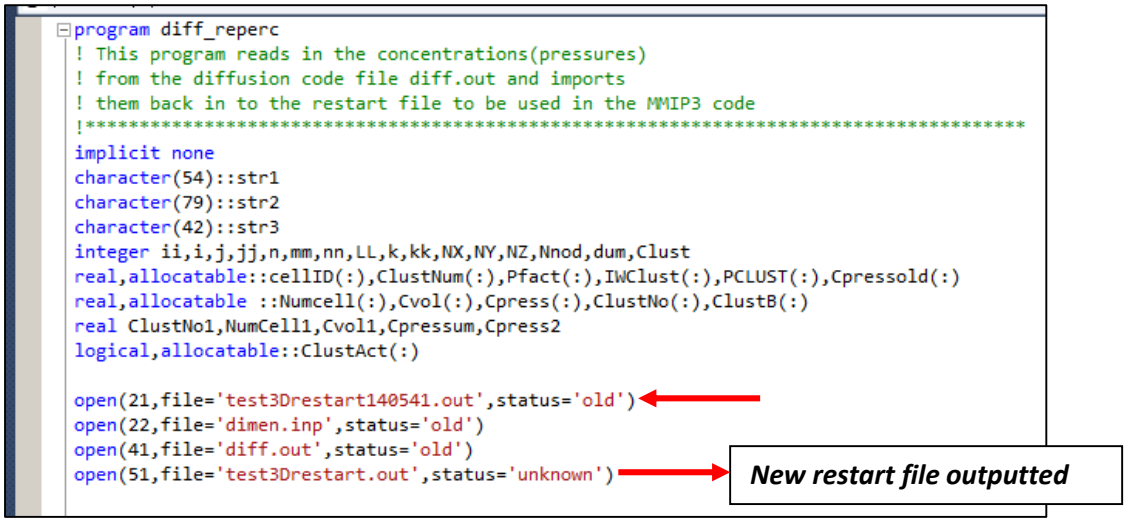



Figure A-20. User input for DIFF_REPERC and DIFF_RESTART. The name of the restart file must be user specified.

### *3.5 Final Simulation: MMIP3*

The code MMIP3 is based on the macro modified invasion percolation method reported by *Glass et al.*, (2001) to simulate the unsaturated non-wetting fluid displacement of a wetting fluid. The MMIP3 code is a descendent of the modified invasion percolation code of *Holt et al.*, (2003) which was used to investigate the influence of centrifugal forces on unsaturated flow in laboratory centrifuge experiments.

MMIP3 is the main program for the modified macroscopic invasion percolation simulator. This code uses pressure formulations within and around the domain boundaries e.g. we assume no flow on all sides except for the top that has an atmospheric pressure of 1.01E5 Pa and overburden pressure of 2.97E5 Pa for 20 m of water. Invasion pressures are determined by the pressure differences across the domain. The program finds all the active nodes that are in contact with the invading fluid (*Holt et al*., 2003). Those nodes are sorted by their invasion pressure and the node with the lowest invasion pressure is invaded first. The process is repeated until equilibrium is attained. The MMIP3 model differs from standard invasion percolation (IP) models through the definition of macro-scale capillarity where individual pore throats and necks are not considered. Instead, a near pore-scale block is defined and characterized by a local threshold spanning pressure or local block scale breakthrough pressure that represents the conditions of the sub grid network. The model domain is discretized in to an array of grid blocks

with assigned spanning pressures from the random field generator SGSIM mentioned previously. An invasion pressure for each block is then determined by summing the spanning pressure, buoyancy forces and viscous forces. An IP algorithm sorts the invadable nodes, selects the node connected to the growing air cluster with the lowest invasion pressure, and invades it. The MMIP3 incorporates a three dimensional clustering algorithm, simultaneous invasion and reinvasion of water and air, hysteris in water and air drainage curves, distributed porosities and drainage parameters, and gas phase compression and trapping. The program structure is beyond the scope of this manual. Additional background about invasion percolation theory and the code structure itself can be found in the MMIP3 Manual included in the program directory.

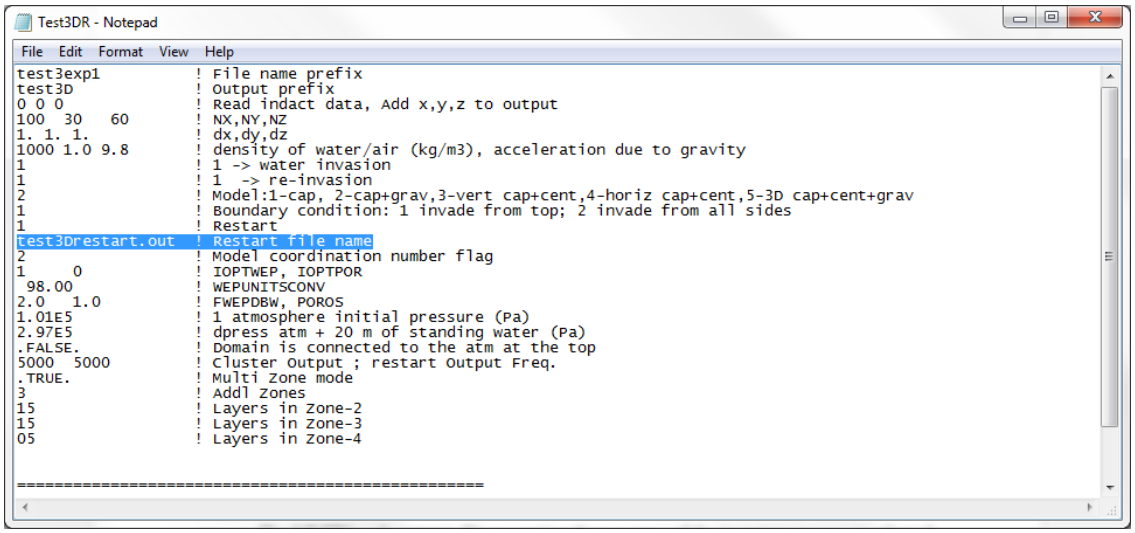


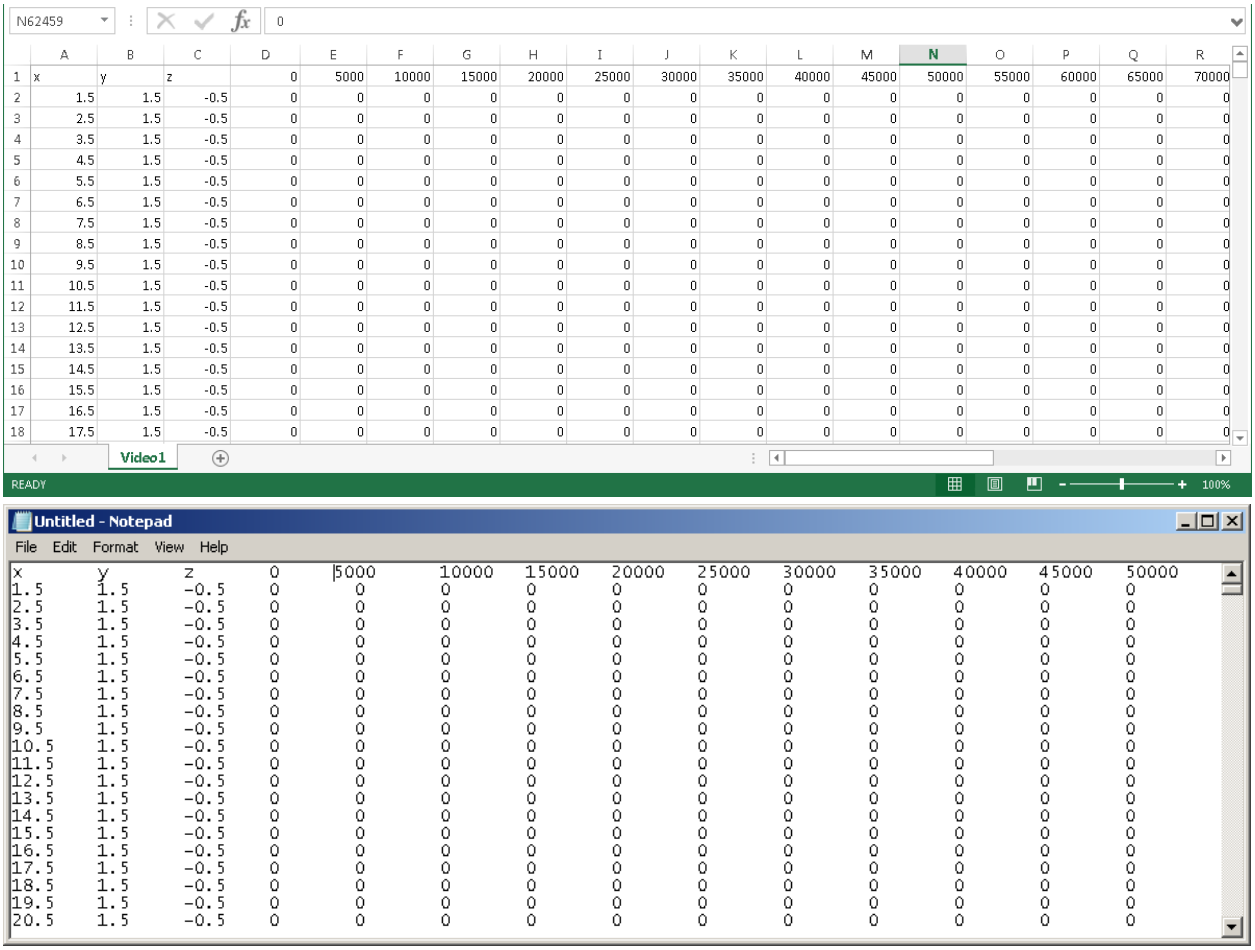

Figure A-21. MMIP3 input file. An example MMIP3 input file with the specified restart file name highlighted.

## 4.0 Visualization

The MMIP3 code outputs files reporting the progress of the invasion process at selected steps, which can then be integrated to display the advance of the invasion front with time, through sequential JPEG images converted in to a video file. The advance and reinvasion of air after the original compilation can be visualized with the images. The output files outstr+clstr+L.out, with different values of L (invasion order step), from 0 to the final step, permit the users to visualize the growth and advance of the invasion front.

### *4.1 mView Software Instructions*

The scientific application software package mView Version 4.0 was used for visualizing the numerical results of the model (*Intera Engineering Ltd., 2006*). The software manual has been supplied in the program directory. The percolation and diffusion programs output files clustr*.out or conc*.out code where * represents the invasion step number or time step. These files should be brought into one file with different columns depicting the state of the front at different invasion steps, using MS-Excel, or a PERL script. The first 3 columns of the input data file need to be the x, y and z, coordinates of each node followed by each invasion or time step in the other columns. After the data has been transferred in to MS-Excel, they should be saved as a formatted text (space delimited) *.prn file (Figure A-22).




| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | x | y | z | 0 | 5000 | 10000 | 15000 | 20000 | 25000 | 30000 | 35000 | 40000 | 45000 | 50000 | 55000 | 60000 | 65000 | 70000 |
| 2 | 1.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 2.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 3.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 4.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 5.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 6.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 7.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 8.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 9.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 10.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 11.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 12.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 13.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 14.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 15.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 16.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 17.5 | 1.5 | -0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
x    y    z    0    5000    10000    15000    20000    25000    30000    35000    40000    45000    50000
1.5  1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
2.5  1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
3.5  1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
4.5  1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
5.5  1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
6.5  1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
7.5  1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
8.5  1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
9.5  1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
10.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
11.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
12.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
13.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
14.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
15.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
16.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
17.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
18.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
19.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
20.5 1.5  -0.5  0    0       0        0        0        0        0        0        0        0        0
```

Figure A-22. Input for mView. Data setup and conversion to text file for inputting in to mView software.

The initial tree set up and object pages that needed to be added can be seen in Figure A-23. All conversion data files must be in the same directory when importing and reading them in. The data file needs to be converted in to .mGEO and .mDAT files by selecting Page>New Utility Page. Select the utility page in the tree then Select Object>New>Generic Grid>Grid Data Conversion on the top menu. The same sequence should be followed for the geometry by selecting Object>New>Generic Grid>Grid Geometry Conversion. The name of the .mGEO and.mDAT files can be named as desired. Next, browse for the .prn data file created from before and select 'Apply' at the bottom right hand side of the screen. The status of the conversion can be seen on the bottom left. Sometimes there may be an error in a line if it is not space delimited etc. After the data has been converted, the grid coordinates and data can be read in. Select Object> New>Scalar Operations>Read Scalar Data Set for the data file. To read the geometry coordinates select Object>New>Read Input>Geometry: 2D/3D Layer Model. Under these newly created layers in the tree, the user must browse for the .mGEO and .mDAT files just created and click apply to read in the data and geometry files. Different columns in the data file may be selected by using Data: Select Scalar.

To plot a 3D visualization, select Page>New Plot Page>3D Spatial. The plot dimensions must be larger than the domain size being used. For example a 100 by 30 by 60 domain size uses the plot dimensions of 600x800 in the 3D-Spatial settings. For the percolation code, the user must define the colors being used by selecting Object>New>Pens &Color Maps>Create Linear Color Map. Two color maps should be created (one for air and one for water). To apply these two colors for air (0s in the .prn file) and water, select Object>New>Data: 3D Color Blocks. There should be two 3D Geometry Color Area/Volume pages in the 3D-Spatial list now. Under 'Color Map' the user will find their newly created linear color maps and can apply different settings for each. The default ColorMap Cold-> Hot system is used for the diffusion files. To view the 3D plot, select Window>3D-Spatial. Additional items may be added to the plot such as a color bar. The object page in the tree must be selected if more items are added (Figure A-23). To view each invasion sequence the user must select the invasion step number under Data::Select Scalar in the Read and Select: Data type list. To plot each image, specify its output location under Output>Bitmap. Select BMP for the file-format and Semi-Auto for the output method. Finally, select Standard at the top of the 3D spatial plot window and select 'Plot Dump'. Remember whenever a setting is changed, the user must hit the 'apply' button on the

bottom right hand side of the interface to activate the change. The sequenced images can then be converted to a video file using any generic video software that reads in multiple images in order. Visualization of the percolation invasion sequence and diffusion process can be seen in Figures A-24 and A-25.

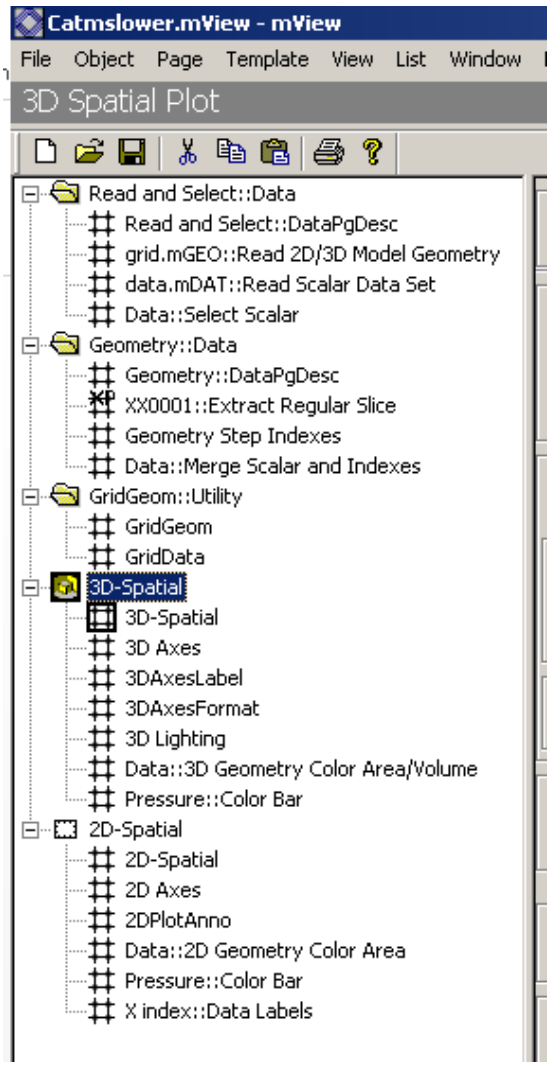



Figure A-23. mView set up. mView settings and graphical interface.

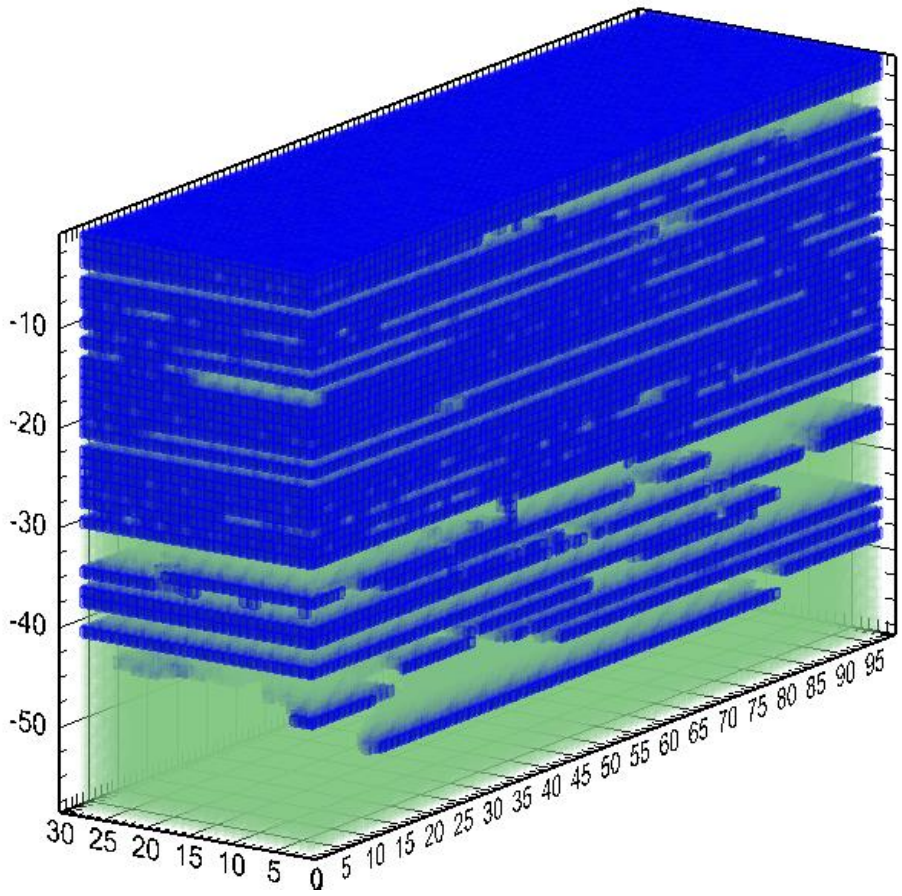



Figure A-24. Percolation invasion. A display of the wetting front (blue) in the invasion percolation sequence produced by mView.

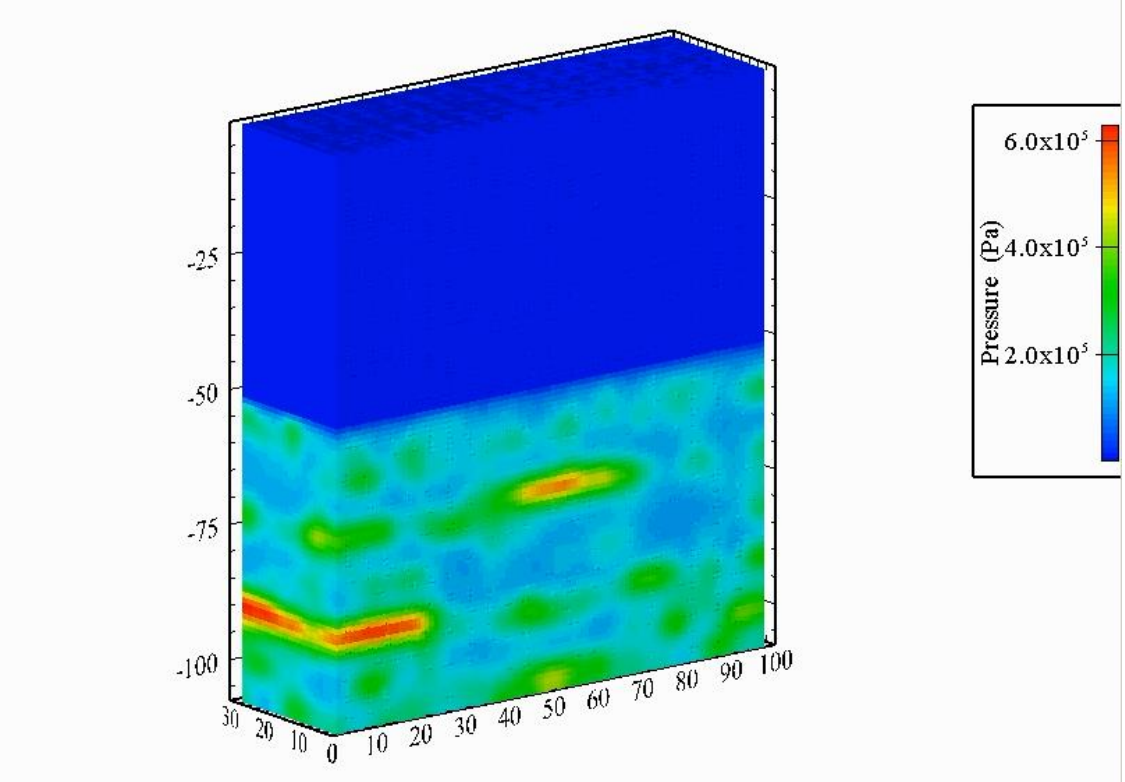



Figure A-25. 3D diffusion example. Visualization of the diffusion code. Blue is water.

## 5.0 Points of Contact

For any questions related to the design and execution of these programs, please contact the following persons. Report the steps you performed when the problem occurs and include the full text of any error messages that are displayed.

**Ross Berry**
Phone: 678-702-3477
Email: raberry@go.olemiss.edu
**Robert M. Holt**
Professor
University of Mississippi, Dept. of Geology/Geological Engineering
120 B Carrier Hall, University, MS 38677-1848
(662) 915-6687
rmholt@olemiss.edu

## 5.0 REFERENCES


Bloomsburg, G.L., and Corey, A.T., 1964, Diffusion of entrapped air from porous media, Hydrology Papers, Colorado State Univ., Fort Collins, CO. No. 5.

Carslaw, H. S., and Jaeger, J. J. C., 1959, Conduction of Heat in Solids, Oxford Science Publications, 2nd edition p. 173.

Crank, J., 1975, The Mathematics of Diffusion, Oxford Science Publications, p. 48.

Deutsch, C.V., Journael, A.G., 1998, GSLIB: Geostatistical software library and users guide. New York, Oxford University Press.

Fry, V.A., Istok, J.D., Semprini, L., Oreilly, K.T., Buscheck, T.E., 1995, Retardation of dissolved oxygen due to a trapped gas phase in porous media, Groundwater, v.33, p. 391-398.

Glass, R.J., Conrad, S.H., and Yarrington, S., 2001, Gravity-destabilized nonwetting phase invasion in macroheterogeneous porous media: Near-pore-scale- macro modified invasion percolation simulation of experiments, Water Resources Research, v. 37, p. 1197-1207.

Golub, G.H., and Van Loan, C.F., 1983, Martrix Computations 4th Edition, p. 625.

Hageman, L.A., and Young, D.M., 1981, Applied iterative methods, Academic Press, New York, Chapter 7.

Holocher, J., Peeters, F., Aeschbach-Hertig, W., Hofer, M., Brennwald, M., Kinzelbach, W., and Kipfer, R., 2002, Experimental investigations on the formation of excess air in quasi-saturated porous media, Geochimica et Cosmochimica Acta, v. 66, p. 4103-4117

Holt, R. M., Glass, R. J., Sigda, J. M., Mattson, E. D.,2003, Influence of centrifugal forces on phase structure in partially saturated media, Geophysical Research Letters, v. 30, NO. 13., 1692.

olt, R.M., E. Hughes, and J. M. Hubbell, 2008, In situ hydrogeologic conditions determined from core samples collected in the vicinity of the Federal Waste Landfill and a playa north of the Federal Waste Landfill at the Waste Control Specialists Site, Andrews County, Texas: Prepared for Waste Control Specialists LLC, P.O. Box 1129, Andrews, Texas.

Holt, R.M., Grisak, G.E., Pickens, J.F., Powers, D.W., Kuszmaul, J., Hughes, E.E., Griffith, C., Cook, S.L., 2010, Conceptual Model Report: submitted as an attachment to a letter from William P. Dornsife, WCS, to Susan Jablonski, Texas Commission on Environmental Quality, on June 24, 2010.

Holt, R.M., Powers, D.W., Hughes, E.E., 2011, Subsurface discontinuity mapping for the federal waste disposal facility and compact waste disposal facility landfills at the Waste Control Specialists Site, Andrew County, Texas: Prepared for Waste Control Specialists LLC, P.O. Box 1129, Andrews, Texas.

IMSL Fortran Numerical Library (FNL) Version 7.0.1 for Intel ® 64, Microsoft Windows 64-bit

Intera Engineering Ltd, 2006, Mview Version 4.0, User Manual, p. 1.

Mcdonald, M.G., and Harbaugh, A.W., 1988, A modular three- dimensional finite-difference ground-water flow model, U.S. Geological Survey, Open-File Report, p. 53.

Navarro, V., Yustres, A., Candel, B., and Garcia, B., 2008, Soil air compression in clays during flood irrigation, European Journal of Soil Science, v. 59, p. 799-806.

**Explanation of Variables**

**Explanation of Variables for DIFF_PERC**

**A** = 2**nx***ny*+1 by *n* array containing the *mxn* positive definite band symmetric coefficient matrix in band symmetric storage mode

**bcF bcB =** boundary condition for front and back

**bcL bcR =** boundary condition for left and right;0=No flow,1=Constant concentration(flux)

**bcT bcBM =** boundary condition for top and bottom

**bvF bvB =** boundary values for front and back

**bvL bvR =** boundary values for left and right

**bvT bvBM =** boundary values for top and bottom

**CBF CBB =** cells that are bounded by a cluster to the front and back, 1=yes, 0=no

**CBL CBR** = cells that are bounded by a cluster cell to the left and right, 1=yes, 0=no

**CBT CBTM =** cells that are bounded by a cluster to the top and bottom, 1=yes, 0=no

**Ccell** = defined array: 1= cell is in cluster, 0 if not; initially defined as **Cindx**

**Cellclust** = number of cells in each cluster; later defined as **Ncell**

**Cindx =** defined array: 1= cell is in cluster, 0 if not

**ClustC =** array of cluster cell concentrations

**ClustN =** array of cluster numbers that coincide to their cell ID

**Cm** = change in mass (total flux into sub-grid domain)

**cndF cndB** = condition for front and back side of sub-grid domain and all nodes

**cndFw cndBw** = condition for front and back side of water domain and all nodes above sub-grid

**cndL cndR** = condition for left and right side of sub-grid domain and all nodes

**cndLw cndRw** = condition for left and right side of water domain and all nodes above sub-grid

**cndT cndBm** = condition for top and bottom side of sub-grid domain and all nodes

**cndTw cndBmw** = condition for top and bottom side of water domain and all nodes above subgrid.

**Cnew** = the new pressure in the cluster updated every time step

**Cold** = previous concentration array to be used for each time step.

**CoutV** = concentrations for visualization output files

**Cu =** unknown concentration array (set to Cold) for PCGRC and final concentration output.

**ddomMS** = mass flux out of sub-grid domain

**Di** = effective diffusion coefficient for sub-grid domain

**domMSold =** initial mass in sub-grid domain

**dt** = time step size

**dtint** = initial time step value later multiplied by tfact

**dv** = volume of each grid block

**Dw** = effective free water diffusion coefficient

**dx** = grid spacing in x direction

**dy** = grid spacing in Y direction

**dz** = grid spacing in Z direction

**f** = load vector

**FF FB** = flux out of the front and back side of cluster cell bounded by a non-cluster cell

**FL FR** = flux out of the left and right side of cluster cell bounded by a non-cluster cell

**Fsum** = the total flux out of a clusters: sum of fluxes in all directions

**FT FBTM** = flux out of the top and bottom side of cluster cell bounded by a non-cluster cell

**i** = grid block index in X direction

**ii** = counter that sums the total number of clusters

**indxF indxB =** index location for front and back of cell m

**indxL indxR =** index location for left and right of cell m

**indxT indxBTM =** index location for top and bottom of cell m

**itmax** = maximum number of iterations

**j** = grid block index in Y direction

**JF JB** = flux out of front and back of sub-grid domain per time step

**JFT JBT** = total flux out of front and back of sub-grid domain per time step

**JL JR** = flux out of left and right side of sub-grid domain per time step

**JLT JRT** = total flux out of left and right side of sub-grid domain per time step

**JT JBM** = flux out of top and bottom of sub-grid domain per time step

**JTT JBMT** = total flux out of top and bottom of sub-grid domain per time step

**k =** grid block index in Z direction

**KH =** Dimensionless Henrys coefficient, dependent on temperature and salinity

**kk =** iterative loop counter for entire domain and number of time steps

**LDA =** leading dimension of matrix *A*

**m =** node id number in vector form (global indexing scheme)

**MassC** = array that calculates the pressure per volume of each cluster

**MassNew=** the change in mass out of each cluster: Mass in – Mass out

**MaxClustCell** = total number of cells that occupy all clusters

**MaxClustN** = number of clusters in domain

**MBE** = mass balance error per time step (Mass in –Mass out; Cm-ddomMS)

**MBET** = cumulative mass balance error

**Mcumerr =** cumulative mass residual error

**Mcumul** = cumulative mass out of the sub-grid domain

**Merror** = residual mass error per time step

**Mint** = initial mass in domain

**Mresid** = fractional mass total out of the sub-grid domain per time step

**MresidT** = fractional mass total remaining in the sub-grid domain per time step

**Murvb =** multiplies the real band matrix in band storage A mode by the load vector f.

**MW =** molecular weight of gas

**Ncell =** number of cells in each cluster

**NMBE** = normalized mass balance error per time step

**NMBET** = cumulative normalized mass balance error

**Nnod =** Total number of nodes without the cluster cells.

**NnodT =** total number of nodes

**Nts** = total number of desired time steps

**NumTS** = number of time steps counter

**Numts** = number of time steps counter for mb_error.f90

**NX** = number of nodes in X direction

**NY** = number of nodes in Y direction

**NZ** = number of nodes in Z direction

**NZD** = number of nodes in Z direction for water overburdern

**PC** = preconditioned matrix in band storage

**Pconv =** array to convert pressures in to concentrations

**phi** = porosity

**Pint** = initial array of pressures may be constant of 1 or 0

**relerr** = relative error in solution

**str** = 5 character name for input and output

**tfact** = time scale factor

**time** = cumulative time counter

**Vxy** = volume of grid blocks in level direction

**Vxz** = volume of grid blocks in vertical direction

**Vyz** = volume of grid blocks in horizontal direction

**Wrrrn** = prints a real rectangular matrix with integer row and column labels

## Explanation of Variables for RANDFIELD

**NX** = number of nodes in X direction

**NY** = number of nodes in Y direction

**NZ** = number of nodes in Z direction

**Nnod =** domain volume NX*NY*NZ

**simnum =** simulation number

**rv =** new random field values calculated

**rvn =** transformed random field value

**meanc =** mean value for clay structure

**means =** mean value for sand structure

## Explanation of Variables for DIFF_RESTART and DIFF_REPERC

**str()** = character names for input and output

**dum** = dummy variables read in; not needed in calculations

**i** = grid block index in X direction

**j** = grid block index in Y direction

**k =** grid block index in Z direction

**kk** = index counter for cluster loop

**ii =** node id number in vector form (global indexing scheme)

**n =** counter index for total number of clusters and invasion step number

**NX** = number of nodes in X direction

**NY** = number of nodes in Y direction

**NZ** = number of nodes in Z direction

**Nnod =** size of domain volume

**Clust** = searches and defines maximum ClustNum for looping and indexing

**cellID =** indexed value ii for cells in domain

**ClustNum =** cluster number

**Pfact =** partial invasion factor for each cluster

**IWClust =** 0 indicates a cell is not trapped; 1 cell is trapped in ClustNum

**PCLUST =** new pressures read in from diff.out

**IP =** cluster pressures multiplied by partial invasion factor of cell

**NumCell =** number of cells in each cluster

**Cvol =** trapped block non-wetting phase volume

**Cpress =** cluster pressure (Pa)

**ClustN =** redefined NumCell

**Ipsum =** sum of the cell pressures in each cluster

**IPP =** next cluster cell pressure added to the sum

**IPNew =** sum of all pressures in cluster divided by number of cells occupied (average)

**IPD =** new array of pressures defined for each cluster

**ClustNo =** cluster number in lower portion of output file

**ClustB =** number of boundary cells in cluster

**ClustAct =** logical variable flag for invadable clusters

**Cpressum =** sum of concentrations in clusters with size > 1

**Cpress =** new averaged cluster pressures based on cluster volume or partial invasion factor

**Cpressold =** initial (old) cluster pressures read in: not applied.

**Cpress2 =** cluster pressures from diff.out in clusters > 1

**Sequential Gaussian Simulation SGSIM**

**Code Structure and Input Variables**

This program requires standard normal data and writes standard normal simulated values. Normal score transforms and back transforms are to be performed outside of this program in the RANDFIELD.f90 program. A Gaussian distribution was used because it is straightforward to establish conditional distributions. The shape of the conditional distributions are Gaussian (normal) and the mean and variance are given by kriging. Summarized steps of SGSIM are given below and referenced straight from the online GSLIB help page.

1. Transform the data to "normal space"
2. Establish a grid network and coordinate system
3. Decide whether to assign data to nearest grid block or keep separate
4. Determine a random path through all of the grid nodes
   a.) Searches for nearby data and previously simulated grid nodes
   b.) Constructs the conditional distribution by kriging
   c.) Draws out a simulation value form the conditional distribution
   d.) Results are back transformed and checked

An example input file is below with the user altered values in bold. Most the input commands in the input file are negligible to the task of this manual and quite extensive. Additional information for the program may be found online or in the GSLIB Geostatistical Software Library Users code book (*Deutsch et al*., 1998). The file with data junk.dat is an absent file that 'triggers' an unconditioned random field. The main variables addressed and changed in this program are nx,ny,nz, maximum search radii, type of covariance structure it, variance cc, and the a_hmax, a_hmin, a_vert variables which are the correlation scale in every direction.

Parameters for SGSIM
**********

START OF PARAMETERS:

| | |
|---|---|
| junk.dat | \file with data |
| 1 2 3 4 0 0 | \columns for X,Y,Z,vr,wt,sec.var. |
| -1.0 1.0e21 | \trimming limits |
| 0 | \transform the data (0=no, 1=yes) |
| sgsim.trn | \file for output trans table |
| 0 | \consider ref. dist (0=no, 1=yes) |
| histsmth.out | \file with ref. dist distribution |
| 1 2 | \columns for vr and wt |
| 0.0 15.0 | \zmin,zmax(tail extrapolation) |
| 1 0.0 | \lower tail option, parameter |
| 1 15.0 | \upper tail option, parameter |
| 1 | \debugging level: 0,1,2,3 |

```
sgsim.dbg                \file for debugging output
sgsim.out                \file for simulation output
1                        \number of realizations to generate
60   0.5   1.0           \nx,xmn,xsiz
30   0.5   1.0           \ny,ymn,ysiz
10   0.5   1.0           \nz,zmn,zsiz
74227                    \random number seed
0    8                   \min and max original data for sim
12                       \number of simulated nodes to use
0                        \assign data to nodes (0=no, 1=yes)
0    3                   \multiple grid search (0=no, 1=yes),num
0                        \maximum data per octant (0=not used)
550 550 20               \maximum search radii (hmax,hmin,vert)
0.0   0.0   0.0          \angles for search ellipsoid
0    0.60   1.0          \ktype: 0=SK,1=OK,2=LVM,3=EXDR,4=COLC
../data/ydata.dat        \file with LVM, EXDR, or COLC variable
4                        \column for secondary variable
1    0                   \nst, nugget effect
1    1.1  0.0   0.0   0.0 \it,cc,ang1,ang2,ang3
540 540 10               \a_hmax, a_hmin, a_vert
```

**Parameters:**

- **datafl:** the input data in a simplified Geo-EAS formatted file. If this file does not exist then an unconditional simulation will be generated.
- **icolx, icoly, icolvr, icolwt** and **icolsec:** the column numbers for the *x*,*y* and *z* coordinates, the variable to be simulated, the declustering weight, and the secondary variable (e.g., for external drift if used). One or two of the coordinate column numbers can be set to zero which indicates that the simulation is 2-D or 1-D. For equal weighting, set **icolwt** to zero.
- **tmin** and **tmax:** all values strictly less than **tmin** and strictly greater than **tmax** are ignored.
- **itrans:** if set to 0 then no transformation will be performed; the variable is assumed already standard normal (the simulation results will also be left unchanged). If **itrans=1**, transformations are performed.
- **transfl:** output file for the transformation table if transformation is required (**igauss**=0).
- **ismooth:** if set to 0, then the data histogram, possibly with declustering weights is used for transformation, if set to 1, then the data are transformed according to the values in another file (perhaps from histogram smoothing).
- **smthfl:** file with the values to use for transformation to normal scores (if **ismooth** is set to 0).
- **icolvr** and **icolwt:** columns in **smthfl** for the variable and the declustering weight (set to 1 and 2 if **smthfl** is the output from histsmth).
- **zmin** and **zmax** the minimum and maximum allowable data values. These are used in the back transformation procedure.

- **ltail** and **ltpar** specify the back transformation implementation in the lower tail of the distribution: **ltail**=1 implements linear interpolation to the lower limit **zmin**, and **ltail**=2 implements power model interpolation, with **w=ltpar**, to the lower limit **zmin**.
- The middle class interpolation is linear.
- **utail** and **utpar** specify the back transformation implementation in the upper tail of the distribution: **utail**=1 implements linear interpolation to the upper limit **zmax**, **utail**=2 implements power model interpolation, with **w=utpar**, to the upper limit **zmax**, and **utail**=4 implements hyperbolic model extrapolation with **w=utpar**. The hyperbolic tail extrapolation is limited by**zmax**.
- **idbg:** an integer debugging level between 0 and 3. The larger the debugging level the more information written out.
- **dbgfl:** the file for the debugging output.
- **outfl:** the output grid is written to this file. The output file will contain the results, cycling fastest on *x* then *y* then *z* then simulation by simulation.
- **nsim:** the number of simulations to generate.
- **nx, xmn, xsiz:** definition of the grid system (*x* axis).
- **ny, ymn, ysiz:** definition of the grid system (*y* axis).
- **nz, zmn, zsiz:** definition of the grid system (*z* axis).
- **seed:** random number seed (a large odd integer).
- **ndmin** and **ndmax:** the minimum and maximum number of original data that should be used to simulate a grid node. If there are fewer than **ndmin** data points the node is not simulated.
- **ncnode:** the maximum number of previously simulated nodes to use for the simulation of another node.
- **sstrat:** if set to 0, the data and previously simulated grid nodes are searched separately: the data are searched with a super block search and the previously simulated nodes are searched with a spiral search (see section II.4). If set to 1, the data are relocated to grid nodes and a spiral search is used and the parameters **ndmin** and **ndmax** are not considered.
- **multgrid:** a multiple grid simulation will be performed if this is set to 1 (otherwise a standard spiral search for previously simulated nodes is considered).
- **nmult:** the number of multiple grid refinements to consider (used only if **multgrid** is set to 1).
- **noct:** the number of original data to use per octant. If this parameter is set less than or equal to 0, then it is not used; otherwise, it overrides the **ndmax** parameter and the data is partitioned into octants and the closest **noct** data in each octant is retained for the simulation of a grid node.
- **radius_hmax**, **radius_hmin** and **radius_vert**: the search radii in the maximum horizontal direction, minimum horizontal direction, and vertical direction (see angles below).
- **sang1, sang2** and **sang3:** the angle parameters that describe the orientation of the search ellipsoid. See the discussion on anisotropy specification associated with Figure II.4.

- **ktype:** the kriging type (0 = simple kriging, 1 = ordinary kriging, 2 = simple kriging with a locally varying mean, 3 = kriging with an external drift, or 4 = collocated cokriging with one secondary variable) used throughout the loop over all nodes. SK is required by theory; only in cases where the number of original data found in the neighborhood is large enough can OK be used without the risk of spreading data values beyond their range of influence
- **rho:** correlation coefficient to use for collocated cokriging (used only if **ktype** = 4).
- **secfl:** the file for the locally varying mean, the external drift variable, or the secondary variable for collocated cokriging (the secondary variable must be gridded at the same resolution as the model being constructed by sgsim).
- **nst** and **c0:** the number of semivariogram structures and the isotropic nugget constant.
- For each of the **nst** nested structures one must define **it**, the type of structure; **cc**, the *c* parameter; **ang1,ang2,ang3**, the angles defining the geometric anisotropy; **aa_hmax**, the maximum horizontal range; **aa_hmin**, the minimum horizontal range; and **aa_vert**, the vertical range.

An example output random field **sgsim.out** and a transformed random field **Test3exp1.rv** from **RANDFIELD** are below. Asteriks * represent the rest of the values down to the end of the output file.

```
 SGSIM Realizations
1
value
          1.1265
          1.1401
          1.3832
          1.3324
          1.2415
          1.2421
          1.2833
          1.2490
         ******
         ******
```

---

```
100         30         30
          17614.89
          17856.09
          22770.00
          21642.19
          19761.67
          19773.53
          20605.23
```

19910.43
20698.15
25139.59
28572.49
24926.81
29554.75
22221.14
********
********

# APPENDIX B: Source Codes and Program Files

Appendix B for this paper is not attached. It includes a copy of the executables and source codes used, user manuals, and all program files.

# VITA

Name: Ross A. Berry

Address: Department of Geology and Geological Engineering
120 Carrier Hall
University, MS 38677

Education: B.S. Geological Engineering, The University of Mississippi, 2012